TUGAS AKHIR – K141502

DETEKSI MOBIL MENGGUNAKAN OPERASI MORFOLOGI DAN KLASIFIKASI ADABOOST

MUHAMMAD AUNORAFIQ MUSA
NRP 5112 100 209

Dosen Pembimbing
Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – K141502

VEHICLE DETECTION USING MORPHOLOGY OPERATION AND ADABOOST CLASSIFICATION

MUHAMMAD AUNORAFIQ MUSA
NRP 5112 100 209

Advisor
Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.
Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

LEMBAR PENGESAHAN

DETEKSI MOBIL MENGGUNAKAN OPERASI MORFOLOGI DAN KLASIFIKASI ADABOOST

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
MUHAMMAD AUNORAFIQ MUSA
NRP : 5112 100 209

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Prof. Dr. Ir. Joko Lianto Buliali, M.Sc. .....
NIP: 19670727 199203 1 003 (pembimbing 1)

Dr.Eng. Chastine Fatichah, S.Kom, M.Kom. .....
NIP: 19751220 200112 2 001 (pembimbing 2)

SURABAYA
MARET 2016

DETEKSI MOBIL MENGGUNAKAN OPERASI MORFOLOGI DAN KLASIFIKASI ADABOOST

Nama Mahasiswa	:	Muhammad Aunorafiq Musa
NRP	:	5112 100 209
Jurusan	:	Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	:	Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.
Dosen Pembimbing 2	:	Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.

ABSTRAK

Pendeteksian mobil memiliki peranan penting dalam manajemen sistem lalu lintas yang dinamis. Hal ini sangat berkaitan dengan penanganan kemacetan di kota-kota besar. Agar penangangan manajemen lalu lintas kendaraan ini bisa optimal, tentu sangat berpengaruh dari informasi yang didapatkan dari sistem pendeteksi kendaraan ini. Hal ini yang menjadi tantangan dalam pendeteksian mobil pada jalan raya maupun jalan bebas hambatan.

Banyak metode yang dapat dilakukan untuk melakukan pendeteksian mobil, mulai dari menggunakan sensor hingga penggunaan kamera CCTV. Sebagai alat yang multifungsi dan lebih murah, pemerintah cenderung menerapkan kamera CCTV sebagai alat yang digunakan untuk memantau mobil di jalan raya maupun di jalan bebas hambatan. Oleh karena itu, diperlukan sebuah sistem deteksi mobil yang menggunakan citra CCTV yang akurat dan mempunyai proses yang cepat.

Metode yang yang digunakan pada tugas akhir ini yaitu melakukan pendeteksian mobil dengan menggunakan dua layer pada tahap pendeteksiannya. Yaitu pada layer pertama yang akan dilakukan operasi morfologi untuk menemukan kemungkinan lokasi lampu depan mobil. Pada layer ini dilakukan operasi Top Hat Transform untuk mendapatkan citra yang kontras, dan metode

Otsu untuk mendapatkan citra biner yang kemudian citra ini dilakukan operasi opening dan closing untuk menghilangkan noise pada citra. Setelah itu dilakukan pencarian lokasi dimana kemungkinan terdapat objek lampu depan dengan menggunakan teknik Connected Component. Hasil dari layer pertama kemudian dilakukan klasifikasi dengan Haar feature based AdaBoost. Klasifikasi ini digunakan untuk mengurangi kejadian false alarm dan meningkatkan nilai precision. Hasil dari metode ini yaitu untuk nilai recall mencapai 91.5% dan untuk nilai precision yaitu 93.8%. Hal ini menandakan jumlah mobil yang tidak terdeteksi dan jumlah kesalahan deteksi pada penggunaan metode ini hanya sedikit.

Kata kunci: ***deteksi mobil, adaptive boosting, operasi morfologi citra.***

VEHICLE DETECTION USING MORPHOLOGY OPERATION AND ADABOOST CLASSIFICATION

Name	:	Muhammad Aunorafiq Musa
NRP	:	5112 100 209
Major	:	Teknik Informatika FTIf-ITS
Supervisor I	:	Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.
Supervisor II	:	Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.

ABSTRACT

Car detection has important role in dynamic traffic management system. It is closely related to the handling of traffic congestion in big cities. In order to optmize the handling of the congestion, the information obtained from this vehicle detection system will be very influential. And this has been the challenge in car detection on highways or freeways.

Many methods can be used to detect car object, starting from the use of sensor and also through CCTV camera. The goverment is likely to implement CCTV used to monitor cars on highways or freeways as multifunctional and cheaper tool. Therefore, a car detection system using CCTV image that is accurate and has rapid process is needed.

In this final project, there are two layers for car detection in the image. The first layer will be conducted using morphological operations to find a possible location of the headlights. In this layer the Top Hat Transfrom operation is conducted to obtain image contrast, and Otsu method to obtain binary image, then this binary image applied Opening and Closing operation to remove the noise. The connected component technique is then applied to the result to find areas where the possible pairs of headlight locate in image. After that, the result from the first layer is conducted using Haar feature based Adaboost. This classification is done to reduce the incidence of false alarm and increase the precision value. The

result show that the method can obtain recall value up to 91.5% and the for the precision value up to 93.8%. This result indicate that the false alarm and missing rate only slightly.

Keywords: ***vehicle detection, adaptive boosting, morphology operation, image processing.***

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

DETEKSI MOBIL MENGGUNAKAN OPERASI MORFOLOGI DAN KLASIFIKASI ADABOOST

Melalui lembar ini, penulis ingin menyampaikan ucapan terimakasih dan penghormatan yang sebesar-besarnya kepada:

1. Bapak, Ibu, adik, kakak dan keluarga besar yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
2. Bapak Prof. Dr. Ir. Joko Lianto Buliali, M.Sc. selaku dosen pembimbing tugas akhir pertama yang telah membimbing dan memberikan banyak masukan dalam pengerjaan tugas akhir ini.
3. Ibu Dr.Eng. Chastine Fatichah, S.Kom, M.Kom. selaku dosen pembimbing tugas akhir kedua yang selalu memberikan koreksi serta banyak masukan-masukan yang dapat penulis kembangkan dari tugas akhir ini
4. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak menyampaikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
5. Sahabat-sahabatku selama kuliah di Jurusan Teknik Informatika ITS, mulai dari kelompok “Sang Pencerah”, administrator Laboratorium Pemrograman 2, dan juga Pengurus Harian HMTC Berkarya yang akan selalu dihati.
6. Teman-teman angkatan 2012 Jurusan Teknik Informatika ITS, khususnya C1C yang telah menjadi teman seperjuangan dalam suka dan duka selama 3 tahun penulis menjalani kuliah.
7. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu per satu.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyusun Tugas Akhir ini, namun penulis mohon maaf apabila terdapat kekurangan yang penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Januari 2016

Muhammad Aunorafiq Musa

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR KODE SUMBER

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Kemacetan saat ini menjadi masalah yang muncul dalam keseharian pada kota-kota besar di Indonesia. Dan hal ini berdampak pada kerusakan lingkungan karena membuat polusi yang banyak dan boros penggunaan energi. Penanganan untuk kemacetan ini adalah dengan membangun jalan alternatif maupun pelebaran jalan. Namun solusi ini sangat sulit untuk direalisasikan karena lahan untuk membangun jalan sangatlah minim, terutama pada kota-kota besar. Terdapat cara lain untuk menangani permasalahan kemacetan ini, yaitu dengan memaksimalkan penggunaan fasilitas *CCTV* atau kamera pada jalan raya. Melalui *CCTV* ini maka akan banyak informasi yang terkumpul mengenai kemacetan di setiap jalan yang kemudian dapat digunakan untuk memanajemen lalu lintas dengan baik.

Deteksi kendaraan melalui citra *CCTV* memiliki peranan yang sangat penting dalam memanajemen sistem lalu lintas yang dinamis. Hal ini sangat berkaitan dengan penanganan kemacetan di kota-kota besar. Penanganan yang cocok dalam manajemen lalu lintas ini tentu sangat berpengaruh dari informasi yang didapatkan dari sistem pendeteksi kendaraan ini.

Yang menjadi tantangan untuk permasalahan ini yaitu berbagai macamnya kondisi lapangan, bisa berupa pencahayaan, tampilan gambar maupun sudut pandang *CCTV*. Sudah ada teknik yaitu *Background Modelling* yang digunakan pada pendeteksian kendaran, namun teknik ini sangat buruk dalam mendeteksi pada kondisi dimana cahaya berubah-ubah. Terlebih untuk pendeteksian

mobil pada malam hari menjadi tantangan karena sulit untuk mengenali bagian dari mobil.

Ada kemudahan dalam pendeteksian kendaraan pada malam hari, yaitu dengan mendeteksi lampu depan kendaraan. Pendeteksian ini menggunakan operasi morfologi pada *layer* pertama, yang operasi ini dilakukan pada gambar yang kemudian diolah menjadi citra biner untuk menentukan lampu sebuah kendaraan. Kemudian, agar tingkat *precision* lebih tinggi, pada *layer* kedua dilakukan operasi *Cascade Classifier* dengan menggunakan AdaBoost sebagai *machine learning* dengan data *training* berupa objek mobil dan objek yang bukan mobil.

Hasil dari aplikasi ini diharapkan dapat menjadi bahan pertimbangan untuk memanajemen lalu lintas dengan cara pengaturan durasi lampu merah.

1.2. Tujuan

Tujuan dari pembuatan tugas akhir ini yaitu membuat program yang mendeteksi kendaraan yang ada pada sebuah *frame video* (*image*) dengan melakukan operasi morfologi dan juga menggunakan AdaBoost sebagai *machine learning* untuk membangun *classifier*. Program dibangun menggunakan pemrograman Java berbasis *dekstop*.

1.3. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini antara lain:

1. Bagaimana cara mendeteksi kendaraan pada malam hari.
2. Bagaimana cara membedakan antara lampu depan mobil dengan lampu motor, lampu jalan, serta pantulan lampu pada jalan.
3. Bagaimana cara mengeliminasi objek yang dapat dikira mobil.
4. Bagaimana cara menerapkan pengolahan citra pada pemrograman java.

1.4. Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, antara lain:

1. Gambar masukan berekstensi jpg dan png.
2. Keluaran dari program ini adalah gambar masukan dengan sudah diberi label mobil.
3. Kendaraan yang dideteksi hanya mobil.
4. Pendeteksian kendaraan hanya berdasarkan pada lampu depan mobil.

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1. **Studi literatur**

   Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu mengenai Java (software platform), operasi Morfologi citra, Haar-Like Feature, Viola And Jones, dan AdaBoost sebagai machine learning.

2. **Implementasi**

   Dalam tugas akhir ini nantinya akan menghasilkan jumlah kendaraan pada gambar *CCTV*. Masukan data berupa gambar *CCTV* jalan raya pada malam hari. Hasil perangkat lunak berupa gambar dengan label kendaraan (bukan sepeda motor) dalam keadaan malam hari. Berikut beberapa hal yang diperlukan dalam implementasi:

   a. IDE Netbeans 8.1
   b. Kemampuan Bahasa Pemrograman Java
   c. Java Development Kit 8.
   d. Pustaka OpenCV 3.0.

3. Pengujian dan evaluasi

Pengujian dari aplikasi ini akan dilakukan dalam cara yaitu :

1. Pengujian blackbox
   Pengujian blackbox adalah pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. Pengujian ini dilakukan untuk menguji apakah posting terkelompokkan ke dalam kategori yang sesuai atau tidak.

4. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan proses dokumentasi dan pembuatan laporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan Tugas Akhir.

1.6. Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Metode Pemecahan Masalah

Bab ini membahas mengenai Metode yang digunakan untuk memecahkan masalah yang dipaparkan pada rumusan permasalahan.

Bab IV Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka aplikasi.

Bab V Implementasi

Bab ini berisi implementasi dari perancangan dan implementasi fitur-fitur penunjang aplikasi.

Bab VI Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode kotak hitam (*black box testing*) untuk mengetahui aspek nilai fungsionalitas dari perangkat lunak dan nilai kegunaan yang dibuat dengan juga memperhatikan ketertarikan pada calon partisipan untuk menggunakan aplikasi ini.

Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir. Teori-teori tersebut meliputi pengertian dan beberapa analisis terkait pendeteksian mobil pada malam hari.

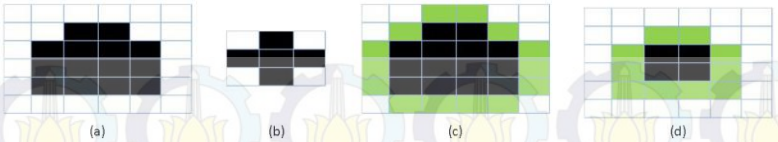
2.1. Morfologi Citra

Operasi morfologi adalah teknik pengolahan citra yang didasarkan pada bentuk segmen atau *region* dalam citra. Karena difokuskan pada bentuk objek, maka operasi ini diterapkan pada citra biner. Biasanya segmentasi tadi didasarkan pada objek yang menjadi perhatian.

Segmentasi dilakukan dengan membedakan antara objek dan latar, antara lain dengan memanfaatkan operasi pengambangan yang mengubah citra warna dan skala keabuan menjadi citra biner. Nilai biner dari citra hasil merepresentasikan 2 keadaan: objek dan bukan objek (latar).

Hasil operasi morfologi dapat dimanfaatkan untuk pengambilan keputusan dengan analisis lebih lanjut. Operasi ini antara lain meliputi : Dilasi, Erosi, penutupan (*closing*) dan pembukaan (*opening*).

Secara umum pemrosesan citra secara morfologi dilakukan dengan cara memberikan sebuah s*tructuring element* terhadap sebuah citra. *Structuring element* dapat diibaratkan seperti *mask* pada pemrosesan citra yang lain selain pemrosesan citra secara morfologi. *Structuring Element* dapat berukuran sembarang, berbentuk mulai kotak hingga lingkaran, dan juga memiliki titik poros [1]. Contoh operasi morfologi dapat dilihat pada Gambar 2.1



Gambar 2.1 Contoh operasi operasi morfologi

Pada Gambar 2.1, bagian (a) merupakan citra masukan, sedangkan bagian (b) merupakan *structuring element* yang digunakan untuk mengatur sensitivitas operasi morfologi terhadap bentuk tertentu (spesifik) pada citra masukan. Operasi dasar dari morfologi yaitu dilasi dan erosi. Dilasi merupakan operasi yang akan menambahan piksel pada batasan dari objek dalam suatu citra sehingga nantinya apabila dilakukan operasi ini maka citra hasilnya lebih besar ukurannya dibandingkan dengan citra aslinya, hasil dari operasi dilasi dapat dilihat pada bagian (c). Sedangkan operasi erosi adalah kebalikan dari operasi dilasi, yaitu membuat ukuran sebuah citra menjadi lebih kecil, contoh hasil operasinya dapat dilihat pada bagian (d). Gabugan kedua operasi ini dapat menjadi operasi *opening* dan *closing*. Persamaan untuk operasi *opening* dan *closing* dapat dilihat pada Persamaan (2.1) dan Persamaan (2.2) secara berurutan

$$A \circ B = (A \ominus B) \oplus B \quad \textbf{(2.1)}$$

$$A \cdot B = (A \oplus B) \ominus B \quad \textbf{(2.2)}$$

Pada persamaan diatas, variabel A merupakan citra masukan, dan variabel B merupakan *structuring element*. Operator $\ominus$ merupakan operasi erosi sedangkan operator $\oplus$ merupakan operasi dilasi.

Operasi morfologi juga dapat diterapkan pada citra *grayscale*. Salah satu kegunaan dari operasi ini yaitu untuk perbaikan citra, atau disebut metode *morphological contrast enhancement*, contoh dari operasi ini yaitu *top-hat transform*. *Top-Hat Transform* atau *Top-Hat by Opening* didefinisikan sebagai perbedaan (*difference*) antara citra input dan hasil *opening* citra

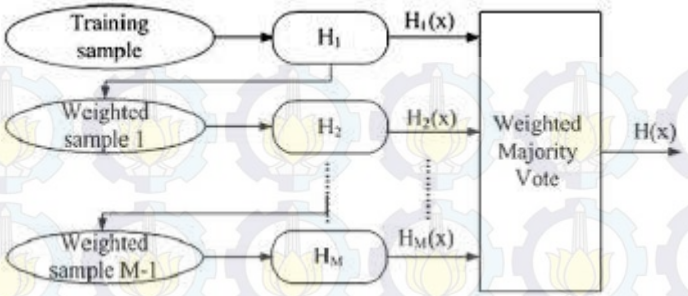
input tersebut oleh suatu *Structuring Element* [2]. Persamaan dari operasi ini dapat dilihat pada Persamaan (2.3). Pada Persamaan tersebut, f merupakan citra masukan, dan $f \circ b$ merupakan hasil *opening* citra dengan *structuring element* b.

$$T_{hat}(f) = f - (f \circ b) \quad \textbf{(2.3)}$$

2.2. AdaBoost (Adaptive Boosting)

Meta-algoritma adalah algoritma yang menggunakan algoritma lain sebagai perwakilan, dan juga merupakan algoritma yang memiliki sub-algoritma sebagai peubah dan parameter yang dapat diganti. Contoh-contoh yang termasuk meta-algoritma adalah *Boosting*.

Boosting merupakan meta-algoritma dalam *machine learning* untuk melakukan *supervised learning*. Secara umum, boosting terjadi dalam iterasi, secara *incremental* menambahkan *weak learner* ke dalam satu *strong learner*. Pada setiap iterasi, satu *weak learner* belajar dari suatu data latihan. Kemudian, *weak learner* itu ditambahkan ke dalam *strong learner*. Setelah *weak learner* ditambahkan, data-data kemudian diubah masing-masing bobotnya. Data-data yang mengalami kesalahan klasifikasi akan mengalami penambahan bobot, dan data-data yang terklasifikasi dengan benar akan mengalami pengurangan bobot [2]. Oleh karena itu, *weak learner* pada iterasi selanjutnya akan lebih terfokus pada data-data yang mengalami kesalahan klasifikasi oleh *weak learner* yang sebelumnya. Penjelasan diatas dapat digambarkan pada Gambar 2.2



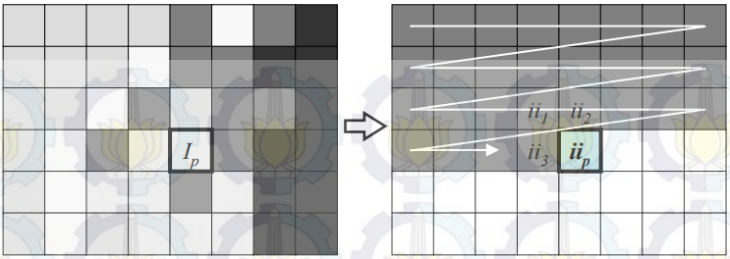
Gambar 2.2 Diagram alir metode Adaptive Boosting

Pada Gambar 2.2, dapat dilihat pada setiap *stages* akan menghasilkan nilai *H* dan berpengaruh pada bobot *stages* selanjutnya.

2.3. Integral Image

Integral Image merupakan sebuah *image* yang nilai tiap pikselnya merupakan akumulasi dari nilai piksel atas dan kirinya [4]. Contoh citra asli dan setelah dikonversi ke *integral image* dapat dilihat pada Gambar 2.3 dan cara perhitungannya dapat dilihat pada Gambar 2.4

Gambar 2.3 Contoh citra masukan dan citra integral

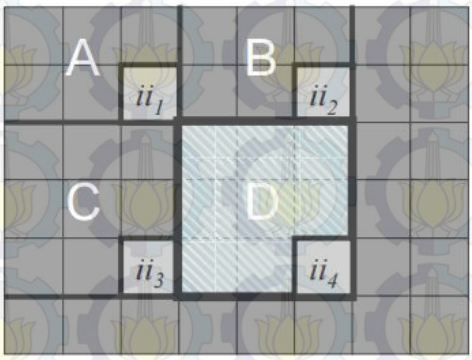


Gambar 2.4 Cara perhitungan *integral image*

Rumus untuk mendapatkan nilai pada titik ii_p dapat dilihat pada persamaan (2.4)

$$ii_p = ii_2 + ii_3 - ii_1 + ii_p \quad \textbf{(2.4)}$$

Sehingga untuk mengambil intensitas dari daerah tertentu pada suatu citra, maka akan digunakan rumus pada persamaan (2.5) sesuai dengan visualisasi pada Gambar 2.5



Gambar 2.5 Contoh *Integral image*

$$Sum_D = ii_4 + ii_1 - ii_2 - ii_3 \quad \textbf{(2.5)}$$

Nilai variable pada persamaan tersebut adalah:

- $ii_1 = sum(A)$
- $ii_2 = sum(A) + sum(b)$

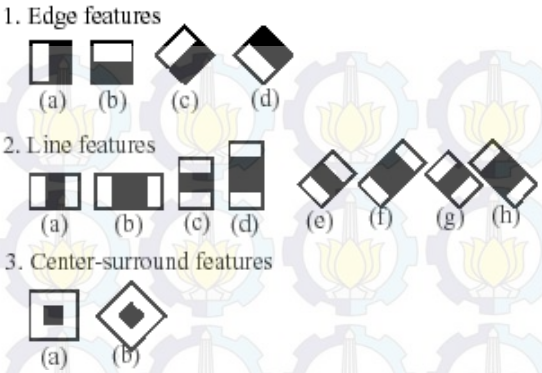
- $ii_3 = sum(A) + sum(C)$
- $ii_4 = sum(A) + sum(B) + sum(C) + sum(D)$

2.4. Haar-like features

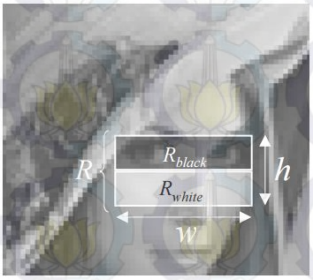
Secara umum, *Haar-Like Feature* digunakan dalam mendeteksi objek pada image digital. Nama Haar merujuk pada suatu fungsi matematika (*Haar Wavelet*) yang berbentuk kotak, prinsipnya sama seperti pada fungsi Fourier. Awalnya pengolahan gambar hanya dengan melihat dari nilai RGB setiap pixel, namun metoda ini ternyata tidaklah efektif. Viola dan Jones kemudian mengembangkannya sehingga terbentuk *Haar-Like Feature*.

Haar-like Feature memproses gambar dalam kotak-kotak, dimana dalam satu kotak terdapat beberapa pixel. Per kotak itu pun kemudian di-proses dan didapatkan perbedaan nilai (threshold) yang menandakan daerah gelap dan terang. Nilai – nilai inilah yang nantinya dijadikan dasar dalam *image processing*.

Lalu untuk gambar bergerak (video), perhitungan dan penjumlahan pixel terjadi secara terus-menerus dan membutuhkan waktu yang lama. Oleh karena itu, penjumlahan diganti dengan integral sehingga didapatkan hasil lebih cepat. Hasil deteksi dari *Haar-Like* kurang akurat jika hanya menggunakan satu fungsi saja sehingga biasanya digunakan beberapa fungsi sekaligus (massal). Semakin banyak fungsi yang digunakan maka hasilnya akan semakin akurat. Pemrosesan *Haar-Like Feature* yang banyak tersebut diatur di dalam *classifier cascade* [3]. Contoh model yang digunakan pada fitur *haar* ini dapat dilihat pada Gambar 2.6. Model pada gambar ini kemudian dihitung nilai intensitas masing-masing kotak, kotak yang hitam maupun yang putih. Visualisasi penerapan kotak *haar-like feature* ini dapat dilihat pada Gambar 2.7 dan perhitungan nilainya menggunakan rumus yang ada pada Persamaan (2.6).



Gambar 2.6 Contoh model fitur *haar-like* [4]



Gambar 2.7 Visualisasi penerapan *haar-like feature* [4]

$$F_{Haar} = E(R_{black}) - E(R_{white}) \quad \textbf{(2.6)}$$

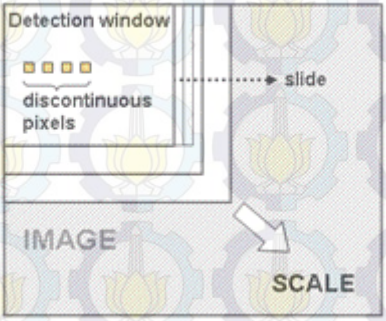
2.5. Metode Viola-Jones

Fitur yang digunakan metode Viola dan Jones menggunakan bentuk *Haar Wavelet*. Bentuk *Haar Wavelet* ialah sebuah gelombang kotak. Pada dua dimensi, gelombang kotak ialah pasangan persegi yang bersebelahan, 1 terang dan 1 gelap. Haar ditentukan oleh pengurangan pixel rata-rata daerah gelap dari pixel rata-rata daerah terang. Jika perbedeaan diatas *threshold* (nilai diatur selama *learning*), fitur tersebut dikatakan ada. Untuk menentukan ada atau tidaknya *Haar-like feature* di setiap lokasi gambar, Viola dan Jones menggunakan teknik yang disebut

Integral Image. Umumnya integral menambahkan unit kecil secara bersamaan. Dalam hal ini unit kecil ini disebut dengan nilai dari pixel. Nilai dari integral *value* pada masing-masing piksel merupakan penjumlahan dari semua pixel di atasnya dan di sebelah kirinya. Dimulai dari kiri atas sampai kanan bawah, gambar dapat diintegrasikan sebagai operasi matematika per pixel.

Untuk memilih *Haar-like feature* yang digunakan dan untuk mengubah nilai *threshold*, Viola dan Jones menggunakan metode *machine learning* yang disebut AdaBoost. AdaBoost menggabungkan banyak *classifier* untuk membuat satu *classifier*. Masing-masing *classifier* menetapkan suatu bobot, dan gabungan dari bobot inilah yang akan membentuk satu *classifier* yang kuat. Viola dan Jones menggabungkan serangkaian AdaBoost classifier sebagai *filter chain* [4].

Metode Viola Jones ini menggunakan *detector* (*detection window*) mulai dari ujung kiri atas gambar, hingga paling kanan bawah. Setelah selesai, selanjutnya *detector* akan diperbesar sesuai dengan skala yang ditentukan. Visualisasi dapat dilihat pada Gambar 2.8

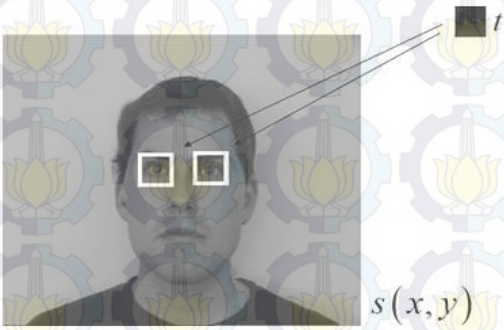


Gambar 2.8 Visualisasi metode Viola and Jones

2.6. Template Matching

Template Matching adalah salah satu teknik yang digunakan untuk mencari area dari sebuah citra yang sesuai dengan citra *template*. Dengan citra sumber (*source image*) sebagai **I** dan

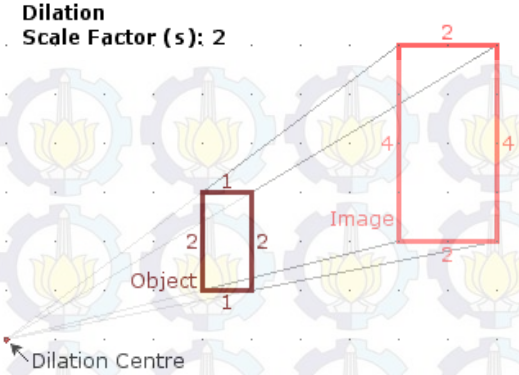
citra *template* sebagai **T.** Metode ini memiliki banyak aplikasi dan digunakan untuk hal seperti pengenalan pola, pengenalan wajah, dan pengolahan citra medis. Metode *Template Matching* akan dilakukan dengan mencocokan citra *template* dengan ROI citra sumber [5]. Salah satu contoh pengaplikasian pada *template* mata pada ROI citra wajah dapat dilihat pada Gambar 2.9



Gambar 2.9 Contoh penggunaan *template matching*

2.7. Scale Factor

Scale Factor adalah nilai perbandingan atau rasio dari 2 bangun geometri yang serupa dan saling berkorespondensi panjang sisi-sisinya [6]. Faktor skala juga bisa diartikan sebagai nilai rasio antara panjang skala gambaran yang berkorespondensi terhadap panjang objek aktual atau nyata.. Contoh dari penggunaan faktor skala dapat dilihat pada Gambar 2.10



Gambar 2.10 Contoh penggunaan faktor skala pada dilasi bangun geometri

2.8. Perhitungan Kinerja Aplikasi

Perhitungan kinerja aplikasi pada sistem ini adalah dengan menggunakan akurasi. Akurasi digunakan untuk mengukur tingkat kualitas deteksi mobil. Kategori uji coba dinyatakan benar jika hasil deteksi adalah mobil. Sedangkan kategori uji coba dinyatakan salah jika hasil deteksi bukanlah objek mobil. Persamaan untuk pengukuran akurasi dapat dilihat pada Persamaan (2.7) dan persamaan (2.8):

$$Recall = \frac{True\ positive}{True\ positive + False\ negative} \quad \textbf{(2.7)}$$

$$Precision = \frac{True\ positive}{True\ positive + False\ positive} \quad \textbf{(2.8)}$$

Pada persamaan diatas, *true positive* pada pengujian ini adalah jumlah mobil yang ada pada citra, *false negative* adalah jumlah kesalahan deteksi yang mobil tapi tidak terdeteksi (*missing rate*), dan *false positive* adalah jumlah kesalahan deteksi yang bukan mobil tapi terdeteksi mobil.

2.9. Haar Training OpenCV

Haar training merupakan pelatiha dengan model statistik *haar* menggunakan beberapa sampel, yaitu sampel positif dan sampel negatif. Contoh dari sampel wajah dapat dilihat pada

Gambar 2.11 Contoh sampel positif dan negatif

Berikut parameter dapat dilihat pada Gambar 2.12 yang digunakan saat pemanggilan fungsi *creatsamples.exe* untuk membuat *file* yang berekstensi *vec* yang akan dijadikan parameter selanjutnya.

```
./createsamples
  [-info <description_file_name>]
  [-img <image_file_name>]
  [-vec <vec_file_name>]
  [-bg <background_file_name>]
  [-num <number_of_samples = 1000>]
  [-bgcolor <background_color = 0>]
  [-inv] [-randinv] [-bgthresh
 <background_color_threshold = 80>]
  [-maxidev <max_intensity_deviation = 40>]
  [-maxxangle <max_x_rotation_angle = 1.100000>]
  [-maxyangle <max_y_rotation_angle = 1.100000>]
  [-maxzangle <max_z_rotation_angle = 0.500000>]
  [-show [<scale = 4.000000>]]
  [-w <sample_width = 24>]
  [-h <sample_height = 24>]
```

Gambar 2.12 Parameter pada *utility creatsamples* OpenCV

Kedua jenis sampel dilatihkan bersamaan dan perbedaannya digunakan sebagai parameter klasifikasi objek terdeteksi wajah atau tidak.

Sampel data yang telah dibuat dilatih menggunakan *haartraining utility*. Berikut ini pada Gambar 2.13 adalah parameter pelatihan data dengan *haartraining utility*

```
Usage: ./haartraining
 -data <dir_name>
 -vec <vec_file_name>
 -bg <background_file_name>
 [-npos <number_of_positive_samples = 2000>]
 [-nneg <number_of_negative_samples = 2000>]
 [-nstages <number_of_stages = 14>]
 [-nsplits <number_of_splits = 1>]
 [-mem <memory_in_MB = 200>]
 [-sym (default)] [-nonsym]
 [-minhitrate <min_hit_rate = 0.995000>]
 [-maxfalsealarm <max_false_alarm_rate = 0.500000>]
 [-weighttrimming <weight_trimming = 0.950000>]
 [-eqw]
 [-mode <BASIC (default) | CORE | ALL>]
 [-w <sample_width = 24>]
 [-h <sample_height = 24>]
 [-bt <DAB | RAB | LB | GAB (default)>]
 [-err <misclass (default) | gini | entropy>]
 [-maxtreesplits
<max_number_of_splits_in_tree_cascade = 0>]
 [-minpos
<min_number_of_positive_samples_per_cluster = 500>]
```

Gambar 2.13 Parameter pada *utility haartraining* OpenCV

Ketika proses *training* sudah selesai makan haartraining akan menghasilkan *file* berekstensi *XML* dan *file* tersebut adalah haarcascade_frontalface_alt.xml. *Haarcascade_frontalface_alt* adalah *training* yang dikhususkan untuk pelacakan wajah dengan posisi geometris lurus ke depan. Berikut ini pada Gambar 2.14 adalah *file* berekstensi *XML* hasil *training*

```
<opencv_storage>
```

```
<haarcascade_frontalface_alt type_id="opencv-haar-classifier">
 <size>20 20</size>
 <stages>
 <_>
 <!-- stage 0 -->
 <trees>
 <_>
 <!-- tree 0 -->
 <_>
 <!-- root node -->
 <feature>
 <rects>
 <_>3 7 14 4 -1.</_>
<_>3 9 14 2 2.</_></rects>
 <tilted>0</tilted></feature>
 <threshold>4.0141958743333817e-003</threshold>
 <left_val>0.0337941907346249</left_val>
 <right_val>0.8378106951713562</right_val></_></_>
 <_>
 <!-- tree 1 -->
 <_>
 <!-- root node -->
 <feature>
 <rects>
 <_>1 2 18 4 -1.</_>
<_>7 2 6 4 3.</_></rects>
 <tilted>0</tilted></feature>
 <threshold>0.0151513395830989</threshold>
 <left_val>0.1514132022857666</left_val>
 <right_val>0.7488812208175659</right_val></_></_>
 <_>
 <!-- tree 2 -->
 ...
<stage_threshold>0.8226894140243530</stage_threshold>
<parent>-1</parent>
<next>-1</next></_>
 <_>
 <!-- stage 1 -->
 ...
```

```
 <!-- stage 21 -->
<trees>
<_>
<!-- tree 0 -->
…
<!-- tree 212 -->
 <stage_threshold>105.7611007690429700</stage_threshold>
 <parent>20</parent>
 <next>-1</next></_></stages></haarcascade_frontalface_alt>
</opencv_storage>
```

Gambar 2.14 Contoh *classifier* hasil training

Training memiliki isi yang berbeda baik dari segi jumlah *stage*, jumlah *tree*, model segi empat dari fitur (*rectangle*) maupun nilai *treshold*-nya. *Stage* melambangkan banyaknya tingkatan dalam *cascade of classifier*, dalam *training* ini digunakan 22 tingkatan (stage 0 sampai stage 21). Tingkatan ini digunakan untuk mengurangi jumlah *sub window* citra yang perlu diperiksa. Di tingkatan pertama dilakukan pengklasifikasian terhadap seluruh sub window citra, lalu di tingkatan kedua dilakukan pengklasifikasian terhadap sub window yang berasal dari hasil pengklarifikasian tingkatan pertama. Maka semakin tinggi tingkatanya semakin sedikiit jumlah sub window yang harus diperiksa. Di tiap tingkatan terdiri dari beberapa tree, biasanya semakin tinggi tingkatan maka tree yang terdapat di dalamnya pun semakin banyak. Pada stage 0 terdapat 2 tree dan pada stage 21 terdapat 212 *tree* [8]. Setelah proses pendeteksian wajah berhasil dilakukan, maka sistem akan menetukan wajah dan tidak. Pada citra yang mengandung wajah dilakukan proses pemberian kotak di sekitar wajah. Hal ini untuk mengecek apakah proses pendeteksian wajah ini telah berhasil mengenali wajah dengan tepat.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada Bab 3 ini akan dijelaskan mengenai analisis dan perancangan perangkat lunak untuk mencapai tujuan dari tugas akhir. Perancangan ini meliputi perancangan data, perancangan proses, dan perancangan antar muka, serta juga akan dijelaskan tentang analisis implementasi metode secara umum pada sistem.

3.1. Analisis Implementasi Metode Secara Umum

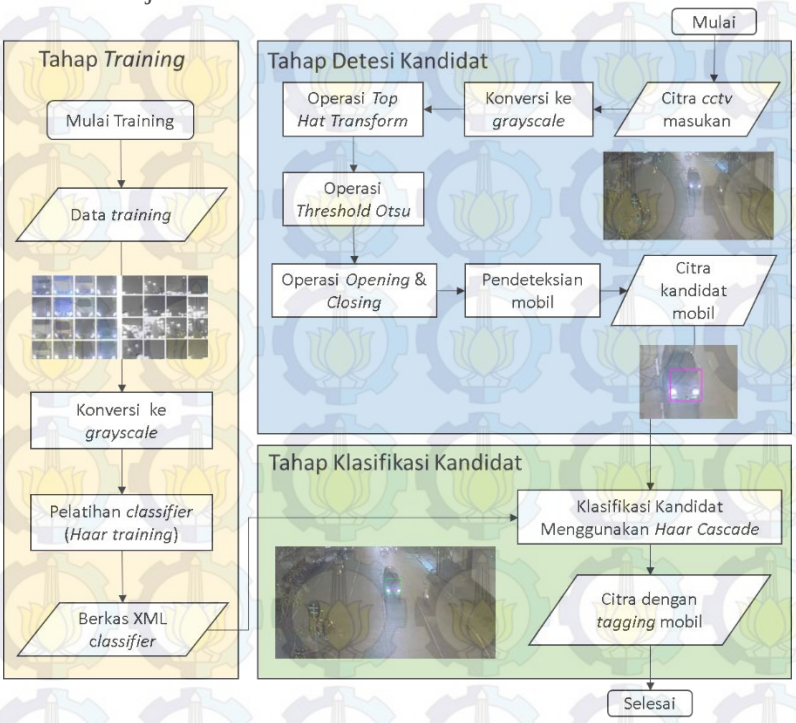
Pada tugas akhir ini akan dibangun sebuah sistem untuk melakukan pendeteksian mobil pada sebuah citra *CCTV* dengan menggunakan pustaka OpenCV. Proses-proses yang terlibat di dalam implementasi sistem ini meliputi tahap *training*, tahap deteksi *area* mobil, dan tahap klasifikasi mobil.

Tahap *training* merupakan tahap mengumpulkan sampel citra *CCTV* mobil bagian depan sebagai sampel positif serta citra yang bukan mobil sebagai sampel negatif untuk kemudian dijadikan sebagai *classifier*. Tahap ini menggunakan *Adaptive Boosting* sebagai algoritma untuk melatih *classifier*. Hasil dari tahap ini berupa *file* dengan ekstensi *XML* (*extended Markup Language*) yang kemudian digunakan pada operasi C*ascade Classifier*. Struktur dari *classifier* ini dapat dilihat pada subbab 2.9.

Tahap deteksi kandidat mobil dilakukan setelah tahap *training* selesai, kedua tahap ini tidak bisa dilakukan secara paralel. Pada tahap ini dilakukan operasi morfologi untuk mencari *area* mobil dengan melacak lampu depan mobil. Tahap ini menghasilkan ROI (*region of interest*) pada citra masukan yaitu berupa kandidat-kandidat objek yang dianggap mobil. Dengan ROI yang didapat pada tahap ini akan mempercepat proses klasifikasi kendaraan. Operasi morfologi yang digunakan yaitu operasi *Top Hat Transform*, operasi ini digunakan untuk membuat kontras antara lampu depan mobil yang terang (*foreground*) dengan objek yang tidak terang (*background*). Setelah itu dilakukan metode *Otsu* untuk mendapatkan citra biner. Citra biner yang dihasilkan dari

metode *otsu* masih terdapat banyak *noise* yang kemudian dilakukan operasi *opening* dan *closing* untuk meminimalisir *noise* tersebut. Sehingga hasil pada operasi *opening* dan *closing* adalah citra biner dengan sedikit *noise*.

Ditahap selanjutnya, dilakukan proses klasifikasi citra menggunakan *Haar Cascade* dengan *classifier* yang sudah dilakukan pada tahap *training* sebelumnya. Tahap ini akan mengeliminasi kandidat yang bukan mobil yang telah dihasilkan pada tahap sebelumnya. Diagram alir dari keseluruhan proses sistem ditunjukkan oleh Gambar 3.1



Gambar 3.1 Gambar diagram alir implementasi metode secara umum

3.2. Perancangan Data

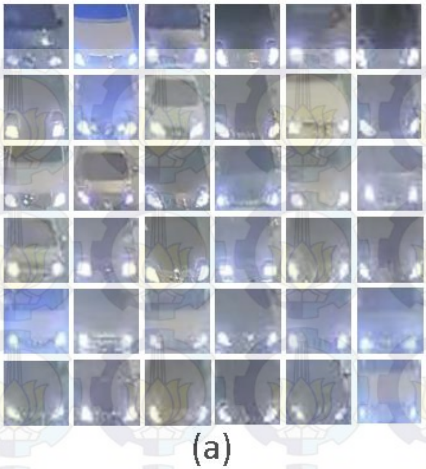
Pada subbab ini akan dibahas mengenai perancangan data yang merupakan bagian penting karena data sebagai objek yang akan diolah oleh perangkat lunak dalam tugas akhir ini dan menghasilkan sebuah informasi. Data yang akan digunakan pada sistem ini adalah data *training*, data masukan (*input*), data proses, dan data keluaran (*output*) yang memberikan hasil pengolahan sistem ini untuk pengguna.

3.2.1. Data Training

Data *training* yang digunakan berupa sampel positif dan juga sampel negatif. Sampel positif berupa citra bagian depan mobil dengan ukuran 24x24 piksel. Sedangkan untuk sampel negatif berupa citra *background* dengan ukuran 24x24, *background* dapat berupa citra yang bukan mobil, bisa seperti jalanan, motor, pohon dsb. Contoh dapat dilihat pada Gambar 3.2

Pembuatan sampel ini dilakukan secara manual dengan cara memotong gambar CCTV yang terdapat bagian depan mobil. Data *training* ini yang kemudian digunakan dalam pencarian fitur *Haar* pada tahap *Training* menggunakan *machine learning* AdaBoost (*Adaptive Boosting*).

Pada Gambar 3.2 ditampilkan 36 dari 538 sampel positif, serta 36 dari 1100 sampel negatif dari keseluruhan data *training* yang digunakan pada sistem deteksi mobil ini.



Gambar 3.2 Contoh sampel positif dan negatif

3.2.2. Data Masukan

Data masukan berupa citra *CCTV* yang didapatkan melalui situs pemerintah surabaya (http://sits.dishub.surabaya.go.id/peta-kepadatan) dengan ukuran 200 x 400 piksel dan memiliki *image*

channel bisa berupa RGB maupun dalam *grayscale*. Contoh citra masukan yang digunakan sebagai data masukan ditunjukkan oleh Gambar 3.3

Gambar 3.3 Citra *CCTV* sebagai data masukan

3.2.3. Data Proses

Data proses adalah data yang digunakan selama proses berjalannya sistem yang merupakan hasil pengolahan dari data masukan untuk diproses kembali menjadi data keluaran di tahap selanjutnya. Data proses yang digunakan dalam proses ini ditunjukkan oleh Tabel 3.1 Data Proses

Tabel 3.1 Data Proses

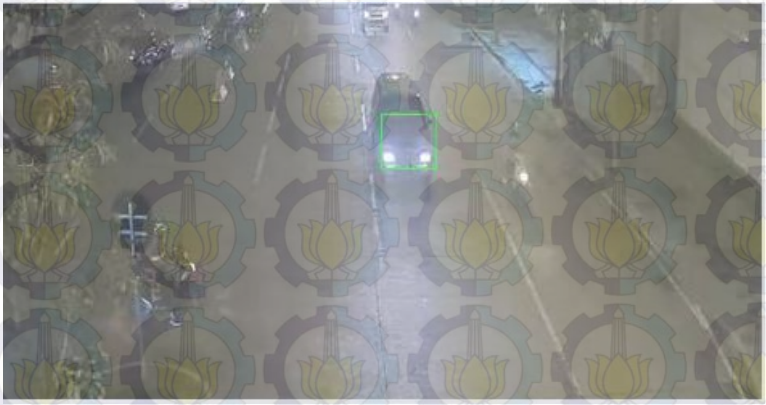
No.	Nama Data	Keterangan
1	fileName	Nama berkas gambar *CCTV* yang akan dijadikan data masukan.
2	fileCascade	Nama berkas *XML* yang akan dijadikan sebagai *cascade classifier*.
3	kernelTophat	Ukuran kernel yang akan digunakan pada operasi *White Top Hat Transform* pada citra masukan.

No.	Nama Data	Keterangan
4	strelMorfologi	Ukuran kernel yang akan digunakan pada operasi *opening* dan *closing* pada citra masukan.
5	DetailObj	Berupa kelas yang berisi atribut detail objek.
6	centroidX	Merupakan atribut dari kelas DetailObj sebagai nilai koordinat sumbu x *centroid* objek.
7	centroidY	Merupakan atribut dari kelas DetailObj sebagai nilai koordinat sumbu y *centroid* objek.
8	mostLeftX	Merupakan atribut dari kelas DetailObj sebagai nilai koordinat sumbu x dari ujung paling kiri atas objek.
9	mostTopY	Merupakan atribut dari kelas DetailObj sebagai nilai koordinat sumbu y dari ujung paling kiri atas objek.
10	width	Merupakan atribut dari kelas DetailObj sebagai nilai lebar dari objek.
11	height	Merupakan atribut dari kelas DetailObj sebagai nilai tinggi dari objek.
12	area	Merupakan atribut dari kelas DetailObj sebagai nilai luas area objek.
13	mOri	Matriks citra masukan awal seperti pada Gambar 3. 2
14	mGrayscale	Matriks citra masukan awal yang telah dikonversi ke *channel grayscale*.
15	mTophat	Matriks citra yang telah dilakukan operasi *White Top Hat Transform* pada citra mGrayscale.

No.	Nama Data	Keterangan
16	mOtsu	Matriks citra yang telah dilakukan operasi *opening* dan *closing* kemudian dilakukan *thresholding* untuk mendapatkan objek.
17	arrRectCrop	Merupakan *array* dari daerah yang dianggap sebagai kandidat mobil.
18	carDetector	Matriks *haar cascade* sebagai *cascade classifier* yang telah dilakukan *training* sebelumnya.
19	kernelTopHat	Ukuran *structuring element* pada operasi *Top Hat*.
20	jumlahObj	Jumlah objek yang ditemukan pada citra biner hasil operasi *opening* dan *dilasi*
21	minLebarDuaLampu	Nilai *threshold* dari lebar ukuran lampu yang menyatu antar lampu kiri dan kanan.
22	maksTinggiDuaLampu	Nilai *threshold* dari tinggi ukuran lampu yang menyatu antar lampu kiri dan kanan.
23	maksUkuranObj	Nilai *threshold* dari maksimal ukuran objek lampu depan.
24	minUkuranObj	Nilai *threshold* dari minimal ukuran objek lampu depan.
25	maksJarakX	Nilai maksimal dari jarak antar lampu pada koordinat *x*.
26	maksJarakY	Nilai maksimal dari jarak antar lampu pada koordinat *y*.
27	batasKiri	Nilai batas kiri pada pencarian objek pada citra.
28	batasAtas	Nilai batas atas pada pencarian objek pada citra.
29	carDetector	*Classifier* yang digunakan pada pencarian *layer* kedua. Variabel ini diisi dengan berkas *XML* hasil *training* menggunakan AdaBoost.

3.2.4. Data Keluaran

Data keluaran dari sistem ini berupa citra masukan dengan tambahan informasi berupa *tag* mobil. Data keluaran dari hasil implementasi sistem ini ditunjukkan pada Gambar 3.4



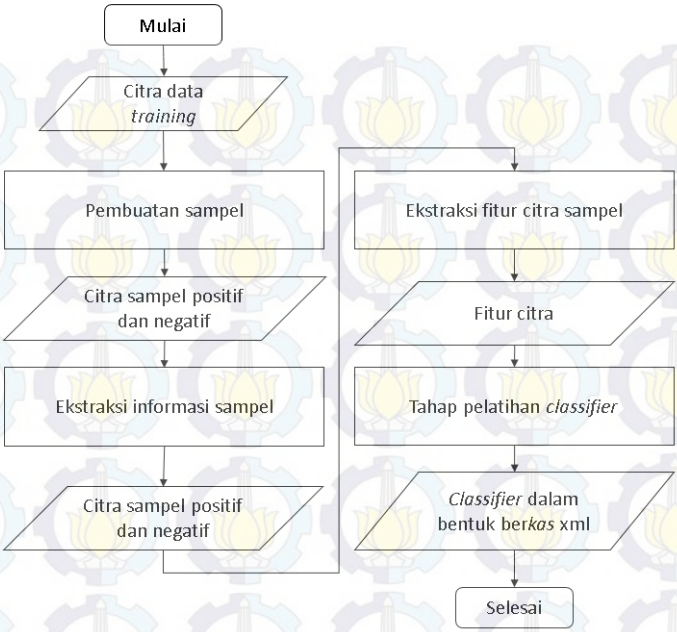
Gambar 3.4 Data keluaran

3.3. Perancangan Proses

Pada subbab ini akan dibahas mengenai perancangan proses yang dilakukan untuk memberikan gambaran secara rinci pada setiap alur implementasi metode pada sistem deteksi mobil. Alur tersebut nantinya akan digunakan dalam tahap implementasi sistem deteksi mobil.

3.3.1. Tahap Training

Pada tahap *training* ini menggunakan pustaka OpenCV 3.0. Terdapat beberapa proses sebelum melakukan *training*, antara lain pembuatan sampel positif maupun negatif, dan ekstraksi informasi sampel. Setelah proses tersebut selesai, kemudian dilanjutkan ke tahap *training classifier* menggunakan fungsi *Haar Training* yang telah disediakan pada pustaka OpenCV [8]. Diagram alir mengenai tahap ini dapat dilihat pada Gambar 3.5



Gambar 3.5 Diagram alir tahap *training classifier*

3.3.1.1. Pembuatan Sampel

Pembuatan sampel dilakukan dengan cara melakukan pemotongan gambar (*cropping*) untuk sampel yang positif maupun yang negatif. Sampel memiliki ukuran yaitu 24x24. Ukuran sampel dibuat seragam agar dalam *training* nanti akan dicari *haar-like feature* yang paling berpengaruh pada ukuran sampel 24x24. Pemilihan ukuran sampel ini dipilih karena ukuran ini sudah bisa memuat fitur-fitur pada bagian depan mobil, seperti lampu depan dan kap depan mobil. Contoh sampel bisa dilihat pada Gambar 3.2.

3.3.1.2. Ekstraksi Informasi Sampel

Pada sampel positif, dibutuhkan sebuah berkas infomasi yang berisi nama citra, jumlah objek pada citra, titik awal objek pada koordinat *x*, dan titik awal objek pada koordinat *y*. Contoh

ekstraksi informasi pada sampel positif seperti pada Gambar 3.6. Sedangkan pada sampel negatif hanya dibutuhkan informasi nama berkas gambar. Hasil dari ekstraksi informasi ini digunakan untuk dijadikan parameter pada fungsi *Haar Training*.

```
rawdata/101a.jpg 1 0 0 24 24
rawdata/102a.jpg 1 0 0 24 24
rawdata/102b.jpg 1 0 0 24 24
rawdata/103a.jpg 1 0 0 24 24
rawdata/104a.jpg 1 0 0 24 24
```

Gambar 3.6 Hasil ekstraksi informasi sampel positif

Pada Gambar 3.6, parameter pertama adalah nama *file* citra dari sampel positif, parameter kedua adalah jumlah objek pada citra, parameter ketiga titik paling kiri dari objek, parameter keempat meruapakan titik paling atas objek, dan parameter kelima dan keenam adalah lebar dan tinggi secara berurutan.

3.3.1.3. Penerapan Algoritma Adaptive Boosting pada Haar Training

Sebelum dilakukan *training*, dilakukan konversi citra dengan *channel* RGB ke *channel grayscale* yang kemudian dilakukan ekstraksi fitur *haar-like* dari citra sampel positif dan negatif. Perhitungan fitur *haar* ini menggunakan *integral image* [3] untuk mempercepat perhitungan nilainya. Rumus *integral image* dapat dilihat pada Persamaan (2.5) untuk mendapatkan nilai *Haar-like Feature*, lebih lengkapnya dapat dilihat pada subbab 2.4.

Setelah ekstraksi fitur *haar-like*, AdaBoost kemudian memilih sejumlah fitur. Pemilihan fitur ini menggunakan *weak learner* dimana setiap iterasinya dilakukan perubahan bobot pada masing-masing *weak learner* [2]. Data-data yang mengalami kesalahan klasifikasi akan mengalami penambahan bobot, dan data-data yang terklasifikasi dengan benar, maka akan dilakukan pengurangan bobot. Oleh karena itu, *weak learner* pada iterasi selanjutnya akan lebih terfokus pada data-data yang mengalami kesalahan klasifikasi oleh *weak learner* sebelumnya.

Untuk masing-masing fitur *weak learner* menentukan *threshold* klasifikasi fungsi yang optimal, sehingga jumlah minimum kesalahan sebuah *weak learner* yang lemah $(h_j(x))$ [5]. Algoritma dari AdaBoost yang diterapan dapat dilihat seperti berikut:

1. Diberikan citra sampel $(x_1, y_1) \dots \dots (x_n, y_n)$ dimana $y_1 \in \{0,1\}$ untuk masing-masing contoh negatif dan positif.
2. Inisialisasi bobot $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ untuk $y_i = 0,1$ secara urut, dimana *m* dan *l* adalah jumlah sampel negatif dan sampel positif secara urut. Variabel $w_{1,i}$ dimana angka 1 merupakan bobot awal, dan variabel i yang menandakan bobot untuk negatif atau positif.
3. *For* $t = 1, \dots, T$
   a. Normalisasi bobot
   $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
   Sehingga w_t adalah nilai distribusi probabilitas untuk setiap *stage* t.
   b. Untuk setiap fitur, *j*, dilatih sebuah *classifier* h_j yang terbatas hanya menggunakan satu fitur saja. Kemudian *error* dievaluasi berdasarkan pada w_t, maka
   $$\epsilon j = \sum_i w_i | h_j(x_i) - y_i |$$
   c. Pilih *classifier*, h_t, dengan tingkat *error* ϵ_t yang paling rendah.
   d. Perbarui bobot:
   $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_t} \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
   Dimana nilai $e = 0$ jika x_i diklasifikasikan dengan benar, $e = 1$ jika sebaliknya. Dan $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

4. Sehingga *Final strong classifier* adalah:

$$h(x)\begin{cases} 1\ jika \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0\ lainnya \end{cases}$$

dimana nilai $\alpha_t = log\frac{1}{\beta_t}s$

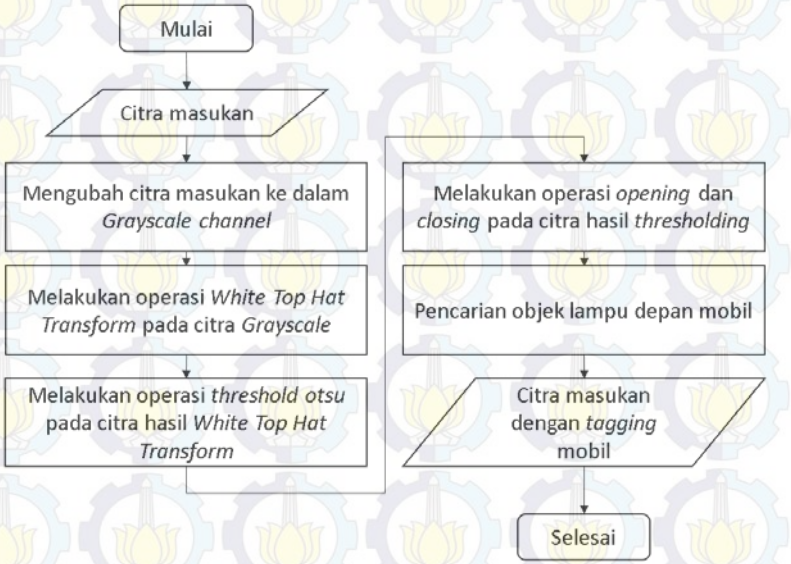
Dan berikut algoritma dari *haar training cascade classifier* yang menggunakan algoritma AdaBoost diatas:

1. Pengguna memilih nilai untuk *f*, yaitu nilai deteksi maksimum yang dapat diterima pertahap, dan nilai *d* sebagai nilai deteksi minimun yang dapat diterima pertahap.
2. Pengguna memilih target keseluruhan dari tingkat kesalahan positif (*false alarm*), F_{target}
3. P = berupa semua sampel positif untuk *training*
4. N = berupa sampel negatif untuk *training*
5. F_0= tingkat *false alarm* awal
6. D_0= tingkat deteksi awal
7. $F_0 = 1{,}0; D_0 = 1{,}0$
8. $i = 0$
9. *While* $F_i > F_{target}$
   a. $i \leftarrow i + 1$
   b. $n_i = 0;\ F_i = F_{i-1}$
   c. *while* $F_i > f \times F_{i-1}$
      i. Gunakan P dan N untuk melatih sebuah *classifier* dengan n_i *features* menggunakan AdaBoost
      ii. Kurangi *threshold* untuk *classifier* ke *i* sampai *cascade classifier* yang sedang berjalan memiliki tingkat deteksi minimal.
   d. $N \leftarrow \emptyset$
   e. Jika $F_i > F_{target}$ maka dilakukan evaluasi *cascade* yang sedang berjalan saat ini pada sampel

negatif, dan menempatkan setiap deteksi kesalahan kedalam *N*.

3.3.2. Tahap Deteksi Area Mobil

Di dalam tahap deteksi *area* mobil, lampu depan mobil menjadi objek yang dicari, terdapat beberapa proses pada tahap ini, yaitu mengubah citra masukan menjadi *grayscale*, penerapan *White Top Hat Transform*, penerapan *Otsu Thresholding*, penerapan operasi morfologi *Opening* dan *Closing* dan pencarian objek lampu depan mobil. Diagram alir mengenai tahap ini dapat dilihat pada Gambar 3.7

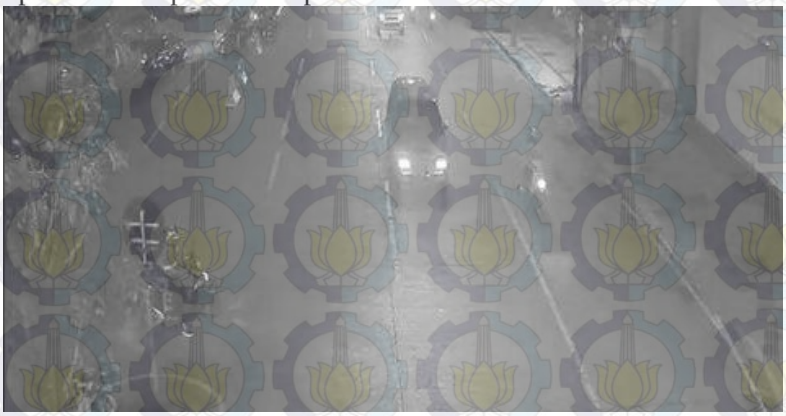


Gambar 3.7 Diagram alir tahap deteksi area mobil

3.3.2.1. Pengubahan Citra ke *Grayscale Channel*

Pengubahan citra masukan ke *grayscale channel* merupakan proses pertama pada tahap pendeteksian mobil. Pemilihan *grayscale channel* ini karena hanya memiliki 1 lapis,

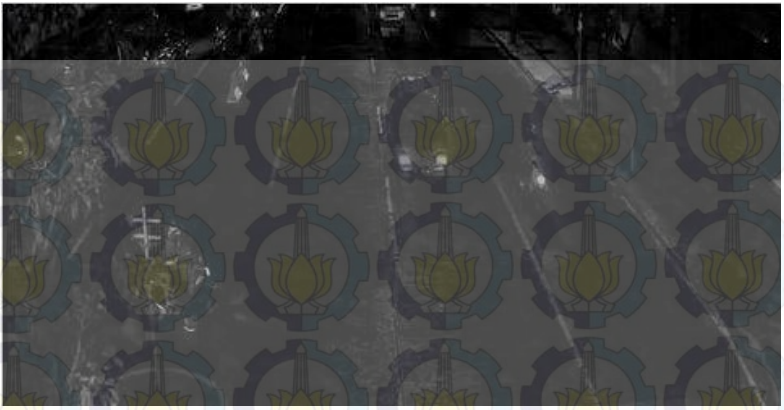
yaitu lapisan *gray*, sehingga dalam melakukan operasi morfologi akan lebih mudah. Selain itu, alasan penggunaan citra *grayscale* karena pada operasi morfologi selanjutnya tidak membutuhkan nilai dari citra yang berwarna. Contoh hasil dari operasi ini dapat dilihat pada Gambar 3.8



Gambar 3.8 Citra hasil operasi *grayscale*

3.3.2.2. Penerapan White Top Hat Transform

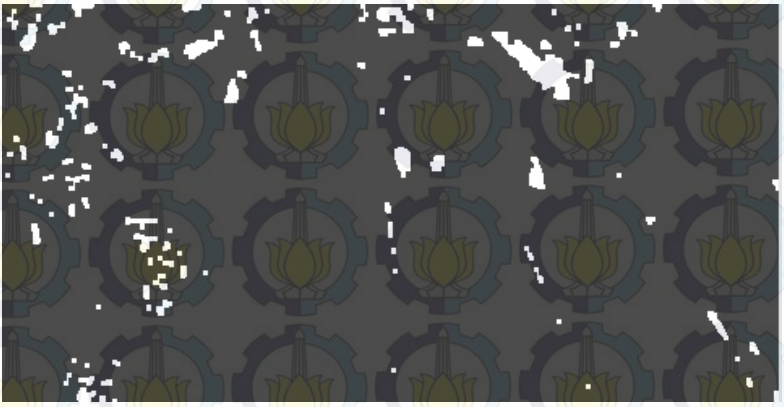
Operasi ini dilakukan untuk mendapatkan citra dengan tingkat kejelasan pada daerah yang menjadi sumber cahaya seperti lampu depan mobil. Pada operasi ini akan mengaburkan bagian yang menjadi pantulan dari cahaya lampu depan mobil. Sehingga operasi ini akan membuat lebih mudah dalam mendeteksi lampu depan mobil. Operasi *Top Hat Transform* ini membutuhkan parameter berupa ukuran dari kernel. Contoh dari operasi ini dapat dilihat pada Gambar 3.9. Untuk melakukan operasi *Top Hat Transform* ini digunakan persamaan (2.3)



Gambar 3.9 Citra hasil operasi *top hat transform* pada citra *grayscale*

3.3.2.3. Penerapan Threshold

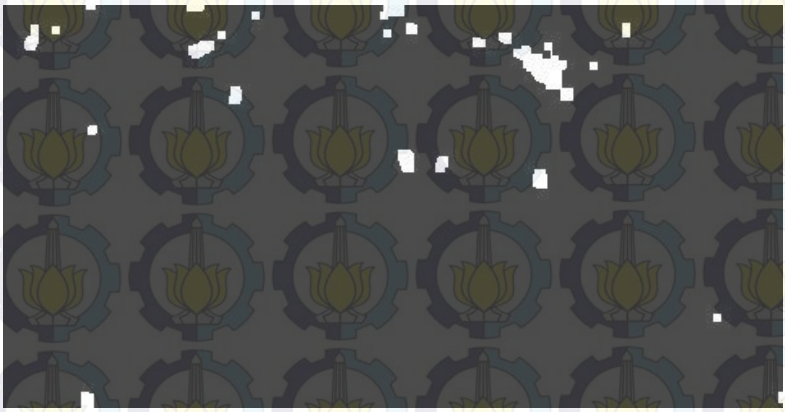
Operasi *Thresholding* citra merupakan proses mengubah citra *grayscale* menjadi citra biner atau hitam putih. Proses ini akan diketahui daerah yang akan menjadi *background* maupun yang menjadi objek. Hasil dari operasi ini dapat diliha pada Gambar 3.10



Gambar 3.10 Citra hasil operasi *thresholding otsu* pada citra hasil *top hat* sebelumnya

3.3.2.4. Penerapan Operasi *Opening* dan *Closing*

Setelah mendapatkan citra biner dari operasi *thresholding*, maka dapat dilihat jika banyak objek *noise* yang muncul. Objek *noise* ini dapat diminimalisir dengan melakukan operasi *opening* kemudian dilanjutkan dengan *closing*. Celah (*gap*) yang terdapat setelah hasil *thresholding* akan hilang sesuai dengan ukuran kernel (*structuring element*). Semakin besar ukuran kernel, maka semua objek akan dikikis dengan objek yang lebih besar. Jika objek lebih kecil dari ukuran kernel, maka objek tersebut akan hilang. Contoh dari operasi ini dapat dilihat pada Gambar 3.11



Gambar 3.11 Citra hasil operasi *opening* dan *closing*

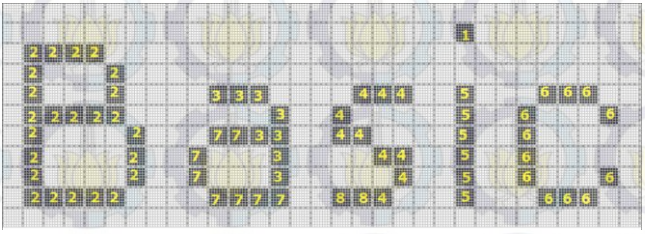
3.3.2.5. Pencarian Objek Mobil

Pada tahap ini, dilakukan pencarian objek dengan menggunakan fungsi *connected component* yang disediakan oleh OpenCV [9]. Fungsi ini hanya dapat dilakukan pada citra biner seperti pada Gambar 3.11 dan fungsi ini dapat dilihat pada persamaan (3.1)

$$S = connectedComponentWithStats\,(src,\ dst,\ stats[\,],\ centroids[\,]) \quad \textbf{(3.1)}$$

Penjelasan dari tiap variabel adalah sebagai berikut:

- *S*: jumlah objek pada citra biner
- *Src*: citra biner masukan
- *dst*: citra keluaran, dapat dilihat pada Gambar 3.12
- *stats*[ ]: *array* dengan isi
  - Koordinat *x* dari titik paling kiri objek
  - Koordinat *y* dari titik paling atas objek
  - Lebar sebuah objek
  - Tinggi sebuah objek
  - Luas area sebuah objek
- *centroids*[ ]: *array* dengan isi
  - Koordinat *x* dari *centroid* objek
  - Koordinat *y* dari *centroid* objek



Gambar 3.12 Hasil operasi *connected Component*

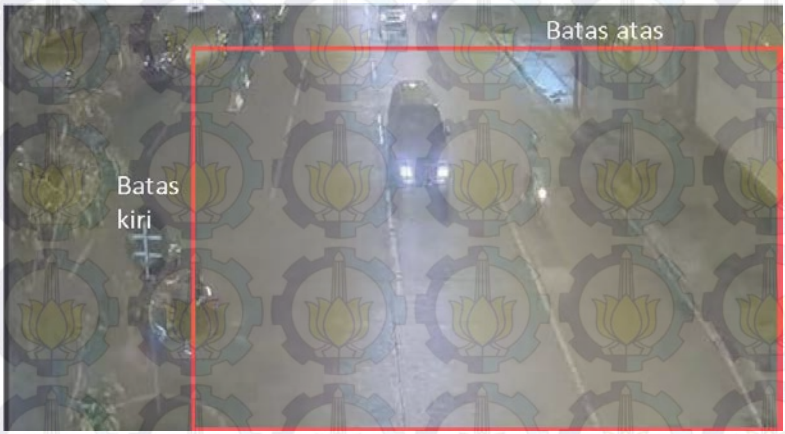
Gambar 3.13 Contoh lampu depan mobil setelah operasi morfologi

Setelah dilakukan fungsi *connected component*, semua variabel *stats*[ ] dan *centroids*[ ] ditampung pada kelas *DetailObj* yang digunakan untuk mencari lampu depan mobil. Setelah kelas *DetailObj* telah diisi, selanjutnya dicari objek yang memungkinkan adalah sebuah lampu depan mobil, contoh lampu

depan mobil setelah operasi morfologi dapat dilihat pada Gambar 3.13. Kriteria *threshold* yang bisa diambil yaitu:

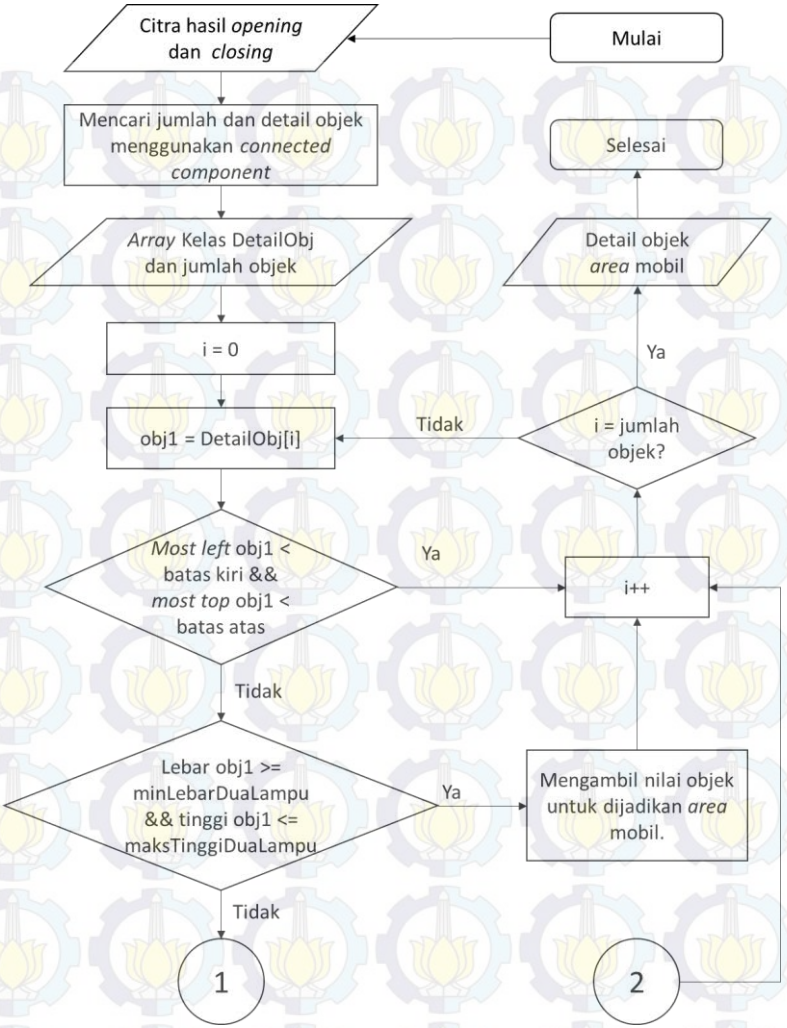
1. Minimal lebar dua lampu
2. Maksimal lebar dua lampu
3. Minimal tinggi dua lampu
4. Maksimal tinggi dua lampu
5. Minimal ukuran lampu
6. Maksimal ukuran lampu
7. Minimal jarak antar lampu
8. Maksimal jarak antar lampu
9. Batas kiri pencarian
10. Batas atas pencarian

Pada *threhold* poin satu hingga empat, digunakan untuk lampu depan yang menyatu antar lampu kiri dan kanan, contoh dapat dilihat pada Gambar 3.13 pada bagian (f). Untuk poin sembilan dan sepuluh digunakan untuk memberikan batasan untuk menghindari daerah yang tidak perlu dilakukan pencarian objek. Daerah pencarian dapat dilihat pada Gambar 3.14

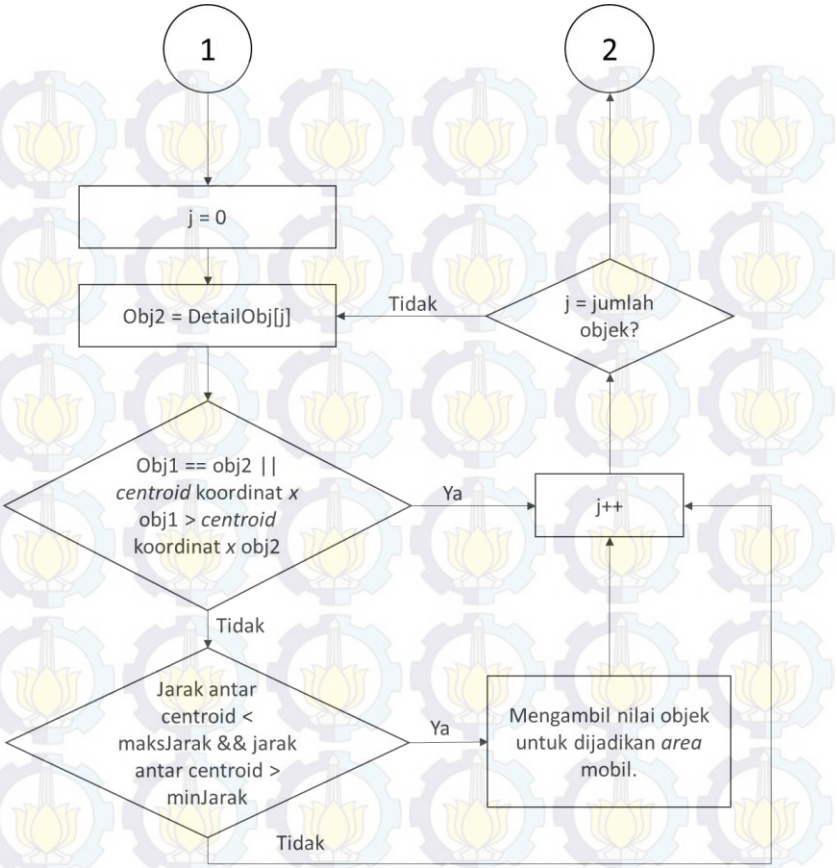


Gambar 3.14 Pemberian batasan pencarian pada citra masukan

Untuk diagram alir dapat dilihat pada Gambar 3.15 dan Gambar 3.16



Gambar 3.15 Diagram alir 1 tahap pencarian area mobil



Gambar 3.16 Diagram alir 2 tahap pencarian area mobil

3.3.3. Tahap Klasifikasi Mobil

Pada tahap ini, *area* mobil yang dihasilnya pada tahap sebelumnya kemudian dilakukan klasifikasi dengan *Haar Cascade*.

3.3.3.1. Klasifikasi menggunakan Haar Cascade

Pada tahap ini, citra masukan sebelumnya akan dilakukan klasifikasi menggunakan *Haar Cascade* untuk memverifikasi

kebenaran kendaraan tersebut. Proses deteksi objek berupa mobil dapat dilihat pada persamaan (3.2):

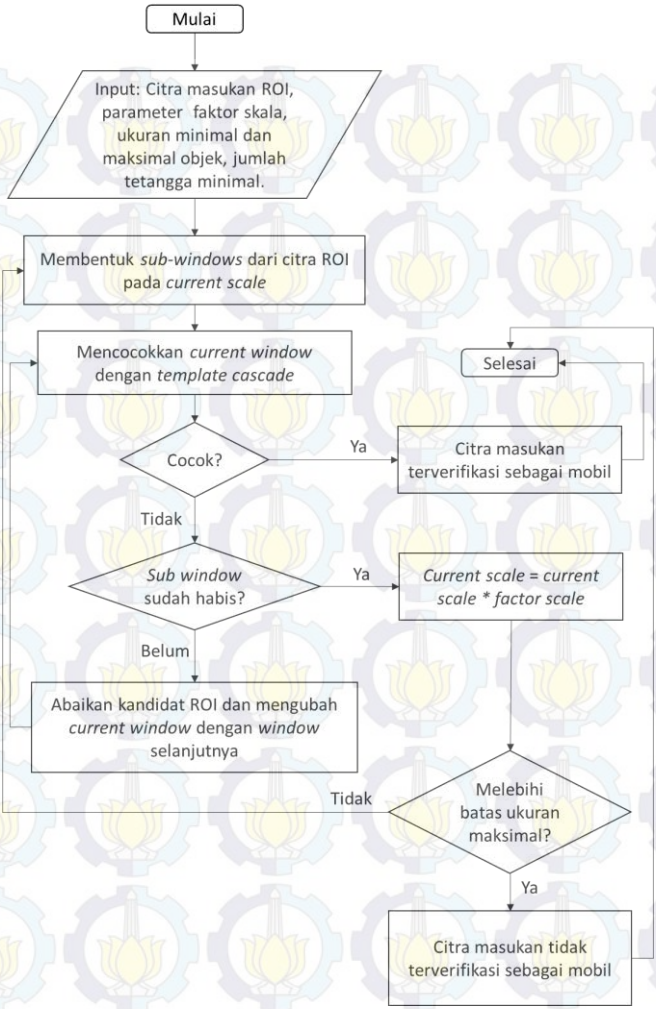
$$R_f[\,] = detectMultiScale(f_g, c_f) \quad (3.2)$$

Variabel f_g adalah citra masukan dan kemudian dilakukan klasifikasi *Haar Cascade* dengan metode *multi scaling* dan menggunakan *cascade template* c_f [9]. Hasil dari operasi ini adalah sebuah *array* lokasi objek yang ditemukan. Klasifikasi *Haar Cascade* secara *multi scaling* adalah proses identifikasi objek yang menggunakan *template* berdasarkan proses *training* sebelumnya.

Terdapat beberapa parameter yang dapat diatur dalam proses pencarian objek menggunakan klasifikasi *Haar Cascade* ini, yaitu:

- *scale faktor* yang menentukan seberapa besar reduksi terhadap citra asli di setiap tingkatan skalanya, karena citra *template* atau sampel *training* tidak selalu memiliki dimensi skala yang sama dengan citra objek pada data *testing*.
- Ukuran minimal *detector*.
- Ukuran maksimal *detector* dari citra objek yang dicari.
- Minimal tetangga yaitu jumlah minimal objek pada daerah tertentu yang terdeteksi benar.

Dengan menggunakan klasifikasi *Haar Cascade* secara *multi scaling* maka dapat ditemukan objek yang dicari sekalipun dengan dimensi data *training* dan data *testing* yang berbeda. Diagram alir pada proses klasifikasi *Haar Cascade* dengan *multi scaling* dapat dilihat pada Gambar 3.17



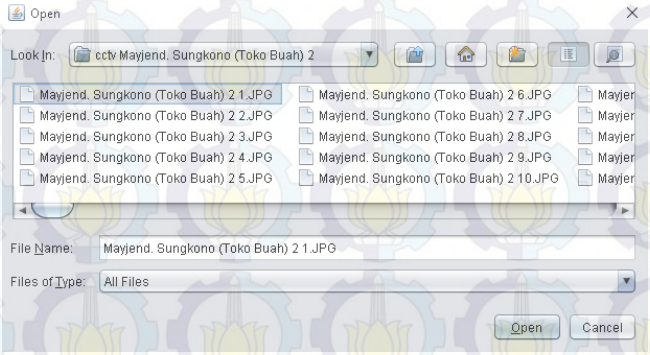
Gambar 3.17 Diagram alir klasifikasi *haarcascade* dengan *multi scaling*

3.4. Perancangan Antarmuka Perangkat Lunak

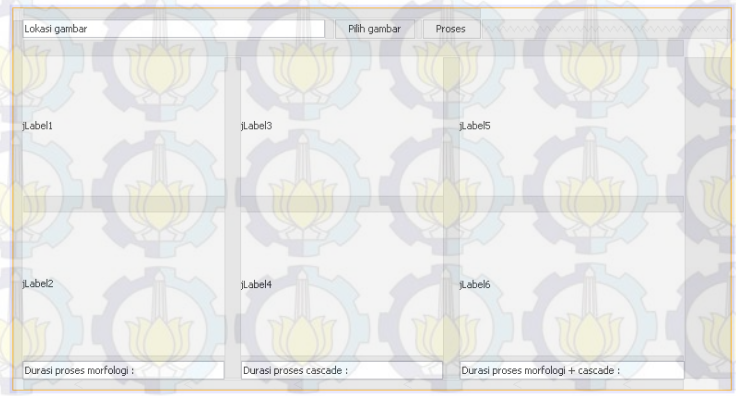
Pada subbab ini akan dibahas mengenai perancangan antarmuka perangkat lunak yang bertujuan untuk dapat

mempermudah interaksi antara perangkat lunak dengan pengguna. Sistem ini hanya memiliki satu halaman yaitu halaman utama.

Halaman utama ini merupakan halaman satu-satunya pada sistem ini. Pada halaman ini pengguna memilih gambar yang akan diproses, proses pemilihan akan menampilkan *widget* seperti pada Gambar 3.18. Setelah tombol proses ditekan, akan ditampilkan enam gambar pada setiap *widget* label yang ada pada Gambar 3.19. Selain itu terdapat juga *widget text area* untuk menampilkan lama proses per metode. Penjelasan mengenai atribut-atribut yang terdapat pada halaman ini dapat dilihat pada Tabel 3.2



Gambar 3.18 *Widget file chooser*



Gambar 3.19 Rancangan halaman utama

Tabel 3.2 Spesifikasi atribut antarmuka halaman utama

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan / Keluaran
1	pilihButton	*Button*	Pilihan aksi untuk menampilkan *widget file chooser*	*Button action*
2	prosesButton	*Button*	Pilihan aksi untuk melakukan proses deteksi mobil	*Button action*
3	fileName	*Text area*	Menampilkan nama *file* yang akan diproses	Nama *file*
4	textMorfologi	*Text area*	Menampilkan durasi proses pada operasi morfologi	Durasi proses
5	textCascade	*Text area*	Menampilkan durasi proses pada operasi *cascade classfier*	Durasi proses
6	textDuaLayer	*Text area*	Menampilkan durasi proses pada operasi morfologi dan *cascade classifier*	Durasi proses
7	jLabel1	*Label*	Menampilkan gambar masukan yang orisinil	*Image*
8	jLabel2	*Label*	Menampilkan hasil proses morfologi	*Image*
9	jLabel3	*Label*	Menampilkan hasil proses hingga tahap *Top Hat Transform*	*Image*
10	jLabel4	*Label*	Menampilkan hasil proses *cascade classifier*	*Image*
11	jLabel5	*Label*	Menampilkan hasil proses hingga tahap *opening* dan *closing*	*Image*
12	jLabel6	*Label*	Menampilkan hasil proses operasi morfologi dan *cascade classifier*	*Image*

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari perancangan sistem sesuai dengan perancangan yang telah dibuat. Bahasa pemrograman yang digunakan untuk implementasi sistem adalah bahasa pemrograman Java dengan pustaka OpenCV 3.0.

4.1. Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir memiliki spesifikasi perangkat keras dan perangkat lunak seperti ditampilkan pada Tabel 4.1

Tabel 4.1 Lingkungan Implementasi Sistem

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i7-3517U CPU @ 1.90GHz (4 CPUs) , ~2.4GHz Memori: 4096 MB
Perangkat lunak	Sistem Operasi: Microsoft Windows 10 64-bit Perangkat Pengembang: Netbeans 8.0.2 Perangkat Pembantu: Notepad++, Microsoft Excel 2013, Microsoft Word 2013

4.2. Implementasi Proses

Implementasi proses dilakukan berdasarkan perancangan proses yang sudah dijelaskan pada bab analisis dan perancangan.

4.2.1. Implementasi Tahap Latihan

Subbab ini membahas mengenai implementasi tahap *training* yang akan digunakan pada tahap selanjutnya. Implementasi tahap ini dilakukan dengan pembuatan sampel, kemudian ekstraksi informasi sampel, penerapan algoritma AdaBoost, dan konversi hasil latihan ke berkas *XML*. Hasil dari

tahap ini adalah berkas *XML* yang merupakan hasil *training* yang akan digunakan sebagai *classifier*. Pada tahap ekstraksi informasi sampel diperlukan fungsi untuk mendapatkan lebar maupun tinggi dari citra, berikut Kode Sumber 4.1 untuk ekstraksi informasi sampel.

```
1  public static void main(String[] args) {
2    File folder = new File(path);
3    File[] listOfFiles = folder.listFiles();
4    String fileName = new String();
5    int sum = 0;
6    String[] output = new String[1000];
7    for(sum=0; sum< listOfFiles.length; sum++){
8        if(listOfFiles[sum].isFile()){
9            fileName =
listOfFiles[sum].getAbsolutePath();
10           Mat m = Imgcodecs.imread(fileName);
11           output[sum] = "rawdata/" +
listOfFiles[sum].getName() + " 1 0 0 " +
m.width() + " " + m.height() ;
12       }
13   }
14 }
```

Kode Sumber 4.1 Kode untuk mengekstraksi informasi sampel positif

Setelah mendapatkan informasi sampel, maka dilakukan pembuatan berkas *vec* yang akan digunakan pada *training*. Kode untuk pembuatan berkas *vec* dapat dilihat pada Kode Sumber 4.2

```
1  createsamples.exe -info positive/info.txt -vec
   vector/carvector.vec -num 532 -w 24 -h 24
```

Kode Sumber 4.2 Kode untuk membuat *vec file*

Parameter pada kode program diatas adalah *–info* sebagai *path* berkas informasi, *-vec* sebagai lokasi file keluaran, *-num* sebagai jumlah data positif, *-w* dan *–h* sebagai lebar dan tinggi dari objek. Untuk menjalankan kode program diatas, maka diperlukan file *creatsamples.exe*, berkas *cv097.dll*, *cxcore097.dll*, *highui097.dll*, dan *libguide40.dll* dalam satu *folder*.

Selanjutnya adalah tahap *training* sekaligus ekstraksi fitur menggunakan program *haartraining.exe*, kode program untuk menjalankan program ini dapat dilihat pada Kode Sumber 4.3

```
1  haartraining.exe -data cascades -vec
   vector/carvector.vec -bg negative/bg.txt -npos
   532 -nneg 1100 -nstages 13 -minhitrate 0.999 -
   maxfalsealarm 0.3 -mem 1024 -mode ALL -w 24 -h
   24 -nonsym
```

Kode Sumber 4.3 Kode program untuk mengeksekusi *haartraining.exe*

Parameter pada kode program diatas yaitu *–data* sebagai lokasi berkas dari *classifier*, *-vec* sebagai lokasi berkas vektor, *-bg* sebagai lokasi berkas negatif sampel, *-npos* sebagai jumlah data positif, *-nneg* sebagai jumlah data negatif, *-nstages* sebagai banyaknya *stage* yang diharapkan, *-minhitrate* sebagai minimal jumlah *hit rate* yang setiap *stage* dapatkan, *-maxfalsealaram* sebagai jumlah maksimum dari *false alarm*, *-mem* sebagai jumlah RAM yang dapat digunakan oleh program. Untuk menjalankan kode program diatas membutuhkan berkas *cv097.dll*, *cxcore097.dll*, dan *highui097.dll*. Untuk informasi lebih detail, dapat dapat dilihat pada subbab 2.9.

Pada tahap selanjutnya adalah mengubah hasil iterasi per *stage* kedalam berkas *XML*, berikut Kode Sumber 4.4 untuk menjalankan program *haarconv.exe*

```
1  haarconv.exe data mynewcardetector16.xml 24 24
```

Kode Sumber 4.4 Kode program untuk menjalankan program *haarconv.exe*

Kode program diatas menghasilkan berkas *XML* yang akan digunakan sebagai *classifier*. Parameter *haarconv.exe* merupakan program yang akan mengkonversi hasil iterasi kedalam berkas *XML*, kemudian data merupakan nama *folder* hasil iterasi yang akan dilanjutkan dengan *string* nama berkas keluaran, dan angka 24 24 merupakan ukuran objek pada saat tahap *training*. Untuk dapat menjalankan perintah ini diperlukan folder yang berisikan hasil iterasi *Adaptive Boosting* yang berupa berkas berekstensi txt, berkas *cv097.dll*, *cxcore097.dll*, dan *haarconv.exe*.

4.2.2. Implementasi Tahap Deteksi Area Mobil

Pada subbab ini akan dilakukan beberapa tahap operasi morfologi, yaitu tahap pengubahan citra ke *grayscale channel*, kemudian penerapan *white top hat transform*, kemudian operasi *threshold*, dilanjutkan operasi *opening* dan *closing*.

Kode program implementasi *grayscale* dapat dilihat pada Kode Sumber 4.5

```
1  Mat mOri = Imgcodecs.imread(fileName);
2  Mat mGrayscale = new Mat(mOri.width(),
   mOri.height(), CvType.CV_8UC1);
3  Imgproc.cvtColor(mOri, mGrayscale,
   Imgproc.COLOR_RGB2GRAY);
4  Mat mOri = Imgcodecs.imread(fileName);
5  Mat mGrayscale = new Mat(mOri.width(),
   mOri.height(), CvType.CV_8UC1);
6  Imgproc.cvtColor(mOri, mGrayscale,
   Imgproc.COLOR_RGB2GRAY);
```

Kode Sumber 4.5 Kode program untuk pengubahan gambar *channel* RGB ke *grayscale*

Setelah menjadi citra *grayscale*, maka dilanjutkan operasi *White Top Hat Transform* dengan kode program yang dapat dilihat pada Kode Sumber 4.6

```
1  public Mat tophatProc(Mat m){
2    Mat tophat = new Mat();
```

```
3   Mat kernel = Mat.ones(kernelTopHat,
    kernelTopHat, CvType.CV_8UC1);
4   Imgproc.morphologyEx(m, tophat,
    Imgproc.MORPH_TOPHAT, kernel);
5   return tophat;
6   }
```

Kode Sumber 4.6 Kode program operasi *Top Hat Transform*

Setelah dilakukan *Top Hat Transform*, maka dilanjutkan operasi *Threshold Otsu* dengan Kode Sumber 4.7

```
1   public Mat thresholdProc(Mat m){
2   Mat otsu = new Mat();
3   Imgproc.threshold(m, otsu, 0, 255,
    Imgproc.THRESH_OTSU);
4   return otsu;
5   }
```

Kode Sumber 4.7 Kode program operasi O*tsu Threshold*

Setelah operasi O*tsu Threshold*, dilakukan *opening* serta *closing* dengan ukuran *structuring* element 5x5 dengan bentuk ellipse. Bentuk *ellipse* dipilih agar bentuk lampu depan tidak hilang. Kode program yang dapat dilihat pada Kode Sumber 4.8

```
1   public Mat openCloseProc(Mat m){
2   Mat ret = new Mat();
3   Size size = new
    Size(kernelDilasi,kernelDilasi);
4   Mat kernelEllipse =
    Imgproc.getStructuringElement(Imgproc.MORPH_ELLIP
    SE, size);
5   Imgproc.morphologyEx(m, m, Imgproc.MORPH_OPEN,
    kernelEllipse);
6   Imgproc.morphologyEx(m, m, Imgproc.MORPH_CLOSE,
    kernelEllipse);
7   return ret;
```

```
8  }
```

Kode Sumber 4.8 Kode program operasi *opening* dan *closing* dengan *structuring element ellipse*

Selanjutnya dilakukan perhitungan objek setelah citra keluaran dari operasi *opening* dan *closing*. Seperti yang dapat dilihat pada Gambar 2.1, objek yang berwarna putih kemudian dicari informasi detailnya mulai dari koordinat *centroid*, koordinat titik awal, lebar, tinggi, dan luas area objek. Informasi detail objek tersebut disimpan pada kelas DetailObj yang dapat dilihat pada Kode Sumber 4.9

```
1  public class DetailObj{
2    public int centroidX=0, centroidY=0,
topLeftX=0, topLeftY=0, width=0, height=0,
area=0;
3    private DetailObj() { }
4    public void setCentroidX(int q){
this.centroidX = q; }
5    public void setCentroidY(int q){ centroidY =
q; }
6    public void setTopLeftX(int q){ topLeftX =
q; }
7    public void setTopLeftY(int q){ topLeftY =
q; }
8    public void setWidth(int q){ width = q; }
9    public void setHeight(int q){ height = q; }
10   public void setArea(int q){ area = q; }
11   public int getCentroidX(){ return centroidX;
}
12   public int getCentroidY(){ return centroidY;
}
13   public int getTopLeftX(){ return topLeftX;
}
14   public int getTopLeftY(){ return topLeftY; }
15   public int getWidth(){ return width; }
```

```
16    public int getHeight(){ return height; }
17    public int getArea(){ return area; }
18  }
```

Kode Sumber 4.9 Kode kelas DetailObj

Untuk mengisi nilai kelas ini dibuat fungsi getDetailObj dengan parameter masukan citra biner pada operasi sebelumnya. Fungsi ini dapat dilihat pada

```
1   public DetailObj[] getDetailObj(Mat m){
2     Mat labelling = m;
3     Mat stats = new Mat();
4     Mat centroids = new Mat();
5     Imgproc.connectedComponentsWithStats(m,
    labelling, stats, centroids);
6     int jlhObj = 0;
7
8     int a1;
9     centroids.convertTo(centroids,
    CvType.CV_32S);
10    int size_centroids = (int)centroids.total()
  * centroids.channels();
11    int[] data_centroids = new
    int[size_centroids];
12    a1 = centroids.get(0, 0, data_centroids);
13    System.out.println(a1);
14
15    jumlahObj = data_centroids.length;
16    DetailObj[] detailObj = new
    DetailObj[jumlahObj];
17    for(int i=0;i<jumlahObj; i++)
18      detailObj[i] = new DetailObj();
```

```
19
20      for(int i=0; i<data_centroids.length;
        i++){
21        if(i%2==1) {
22          detailObj[jlhObj].setCentroidY(
        data_centroids[i] );
23            jlhObj++;
24          } else detailObj[jlhObj].setCentroidX(
        data_centroids[i] );
25      }
26
27      int size_stats = (int)stats.total() *
        stats.channels();
28      int[] data_stats = new int[size_stats];
29      int[][] dataStats2 = new
        int[size_stats/5][size_stats];
30      a1 = stats.get(0, 0, data_stats);
31      int y=0, jlhObj2=0;
32      for(int i=0; i<data_stats.length; i++){
33        switch (i%5) {
34          case 0:
35            detailObj[jlhObj2].setTopLeftX(
        data_stats[i] ); break;
36          case 1:
37            detailObj[jlhObj2].setTopLeftY(
        data_stats[i] );
38      break;
39          case 2:
40            detailObj[jlhObj2].setWidth(
        data_stats[i] );
41      break;
42          case 3:
43            detailObj[jlhObj2].setHeight(
        data_stats[i] );
44      break;
```

```
45          case 4:
46            detailObj[jlhObj2].setArea(
data_stats[i] );
47            jlhObj2++;
48  break;
49          default: break;
50          }
51      }
52      return detailObj;
53  }
```

Kode Sumber 4.10 Fungsi untuk mengisi kelas DetailObj

Setelah mendapatkan informasi detail dari objek, kemudian dilakukan pencarian objek yang memungkinkan merupakan mobil. Fungsi pencarian objek mobil ini dapat dilihat pada Kode Sumber 4.11

```
1   public Rect[] kandidatLayer1(DetailObj[]
    detailObj, Mat m){
2     Rect[] arrRectCrop = new Rect[10000];
3     int batasBawah, batasKanan, widthRoi,
    heightRoi, xRoi, yRoi;
4     int thresholdDuaLampu = 0;
5     int thresholdJarakX = 0;
6     for(int i=1;i<jumlahObj;i++){
7       if(detailObj[i].getTopLeftY() < (
    m.height() / 2)) {
8         thresholdDuaLampu = minLebarDuaLampu;
9         thresholdJarakX = minJarakX;
10      }
11      else {
12        thresholdDuaLampu = minLebarDuaLampu2;
13        thresholdJarakX = minJarakX2;
```

```
14      }
15      if(detailObj[i].getTopLeftX() < batasKiri
|| detailObj[i].getTopLeftY() < batasAtas)
continue;
16      if(detailObj[i].getWidth() >=
thresholdDuaLampu && detailObj[i].getHeight()
<= maksTinggiDuaLampu){
17          batasBawah = detailObj[i].getTopLeftY()
+ detailObj[i].getHeight();
18          batasKanan = detailObj[i].getTopLeftX()
+ detailObj[i].getWidth();
19
20          widthRoi = detailObj[i].getWidth();
21          heightRoi = detailObj[i].getHeight();
22          xRoi = detailObj[i].getTopLeftX();
23          yRoi = batasBawah - widthRoi;
24          if(yRoi < 0) {
25            continue;
26          }
27          int xRoi2 = xRoi -
(widthRoi*persentase/100);
28          int yRoi2 = yRoi -
(widthRoi*persentase/100);
29          int widthRoi2 = widthRoi + (
(widthRoi*persentase/100) * 2);
30          int heightRoi2 = heightRoi + (
(widthRoi*persentase/100) * 2);
31          if(xRoi2 < 0 && yRoi2 < 0 && widthRoi2 <
0 && heightRoi2 < 0 ) continue;
32
33          if(xRoi2 <=0){
34            xRoi2 = 0;
35          }
36          if(yRoi2 <= 0){
37            yRoi2 = 0;
```

```
38          }
39          if(xRoi2 + widthRoi2 > m.width()){
40            widthRoi2 = m.width() - xRoi2;
41          }
42          if(yRoi2 + heightRoi2 > m.height()){
43            heightRoi2 = m.height() - yRoi2;
44          }
45          Imgproc.rectangle(m, new Point(xRoi,
yRoi), new Point(batasKanan, batasBawah), new
Scalar(255, 0, 255), 1);
46          Rect tempCrop = new Rect(xRoi2, yRoi2,
widthRoi2, heightRoi2);
47          arrRectCrop[jlhKandidat] = tempCrop;
48          jlhKandidat++;
49        }
50      for(int j=1;j<jumlahObj;j++){
51          int jarakX = detailObj[i].getCentroidX()
- detailObj[j].getCentroidX();
52          int jarakY = detailObj[j].getCentroidY()
- detailObj[i].getCentroidY();
53
54          if(i==j || detailObj[i].getCentroidX() >
detailObj[j].getCentroidX()
55            || detailObj[i].getArea() >
maksUkuranObj
56            || detailObj[j].getArea() >
maksUkuranObj
57            || detailObj[j].getArea() <
minUkuranObj
58            || detailObj[i].getArea() <
minUkuranObj) continue;
59
60          if(Math.abs(jarakX) < maksJarakX &&
Math.abs(jarakY) < maksJarakY &&
Math.abs(jarakX) > thresholdJarakX){
```

```
61          int titikTerendahI = detailObj[i].getTopLeftY() + detailObj[i].getHeight();
62          int titikTerendahJ = detailObj[j].getTopLeftY() + detailObj[j].getHeight();
63          batasBawah = min(titikTerendahI, titikTerendahJ);
64          batasKanan = detailObj[j].getTopLeftX() + detailObj[j].getWidth();
65
66          widthRoi = batasKanan - detailObj[i].getTopLeftX();
67          heightRoi = widthRoi ;
68          xRoi = detailObj[i].getTopLeftX();
69          yRoi = batasBawah - widthRoi;
70          if(yRoi < 0) break;
71
72          Imgproc.rectangle(m, new Point(xRoi, yRoi),  new Point(batasKanan, batasBawah), new Scalar(255, 0, 255), 1);
73
74          int xRoi2 = xRoi - (widthRoi*persentase/100);
75          int yRoi2 = yRoi - (widthRoi*persentase/100);
76          int widthRoi2 = widthRoi + (widthRoi*persentase/100) * 2);
77          int heightRoi2 = heightRoi + (widthRoi*persentase/100) * 2);
78          if(xRoi2 == 0 || yRoi2 == 0 || widthRoi2 == 0 || heightRoi2 == 0) continue;
79
80          if(xRoi2 < 0){
81            xRoi2 = 0;
82          }
```

```
83              if(yRoi2 < 0){
84                yRoi2 = 0;
85              }
86              if(xRoi2 + widthRoi2 > m.width()){
87                widthRoi2 = m.width() - xRoi2;
88              }
89              if(yRoi2 + heightRoi2 > m.height()){
90                System.out.println("qwqwq");
91                heightRoi2 = m.height() - yRoi2;
92              }
93              Rect tempCrop = new Rect(xRoi2, yRoi2,
    widthRoi2, heightRoi2);
94              arrRectCrop[jlhKandidat] = tempCrop;
95              jlhKandidat++;
96            }
97          }
98        }
99        jLabel2.setIcon(new
    ImageIcon(toBufferedImage(m)));
100       return arrRectCrop;
101   }
```

Kode Sumber 4.11 Fungsi pencarian objek mobil

Fungsi kandidatLayer1 ini memiliki *return value* berupa tipe data *array* dari *Rect* yang merupakan *area* yand diduga ada objek mobil. *Area* ini yang kemudian menjadi masukan pada tahap klasifikasi mobil selanjutnya.

4.2.3. Implementasi Tahap Klasifikasi Mobil

Pada tahap ini akan dilakukan klasifikasi dengan *Haar Cascade* untuk memverifikasi kandidat object mobil. Proses ini menggunakan *template* yang merupakan hasil dari *training* pada

tahap sebelumnya. Proses verifikasi menggunakan *haar cascade* dengan *multi scaling* yang telah disediakan pada pustaka OpenCV. Implementasi proses ini pada sistem dapat dilihat pada Kode Sumber 4.12

```
1  public boolean verifikasiKandidat(Mat m){
2    CascadeClassifier carDetector = new
CascadeClassifier(fileCascade);
3    MatOfRect carDetections = new MatOfRect();
4    carDetector.detectMultiScale(m, carDetections,
1.1, 1, 0, new Size(30,30), new Size(50,50) );
5    if(carDetections.toArray().length != 0) {
6      return true;
7    }
8    return false;
9  }
```

Kode Sumber 4.12 Fungsi verifikasi kandidat

4.3. Implementasi Antarmuka Perangkat Lunak

Implementasi tampilan antarmuka pengguna pada perangkat lunak berbasis *dekstop* dan berjalan pada sistem operasi Windows 10. Implementasi menggunakan *GUI Editor* bawaan dari IDE Netbeans 8.0.2. Hanya terdapat satu halaman untuk Sistem Deteksi ini, yaitu halaman utama.

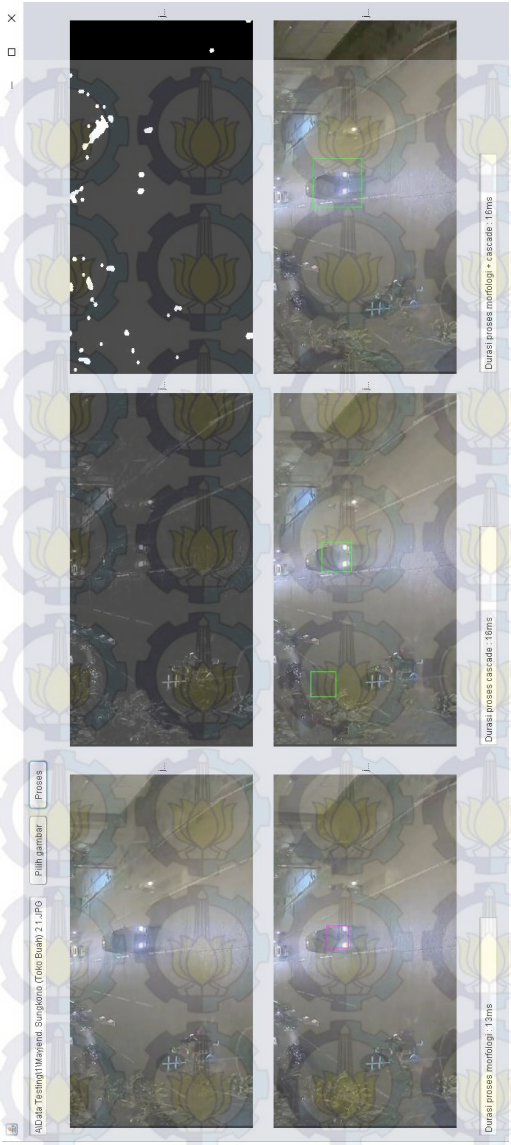
Halaman utama ini digunakan untuk menampilkan semua hasil proses pada sistem ini. Rancangan mengenai halaman ini dapat dilihat pada subbab 3.4 dan implementasinya dapat dilihat pada Gambar 4.1, sedangkan kode programnya dapat dilihat pada Kode Sumber 4. 14 pada lampiran. Pada gambar terlihat bahwa halaman terdapat dua tombol, yaitu tombol pilih yang akan menampilkan *widget file chooser* seperti pada Gambar 3.18 serta tombol proses yang akan memanggil fungsi yang ada pada Kode Sumber 4.13

```
1  private void prosesButtonActionPerformed(java.awt.event.ActionEvent evt) {
2    filenameTextField.setText(fileName);
3    jumlahObj = 0;
4    jlhKandidat = 0;
5
6    Mat mOri = Imgcodecs.imread(fileName);
7    jLabel1.setIcon(new ImageIcon(toBufferedImage(mOri)));
8
9    Mat mGrayscale = new Mat(mOri.width(), mOri.height(), CvType.CV_8UC1);
10   Imgproc.cvtColor(mOri, mGrayscale, Imgproc.COLOR_RGB2GRAY);
11
12   Mat mTophat = tophatProc(mGrayscale);
13   jLabel3.setIcon(new ImageIcon(toBufferedImage(mTophat)));
14
15   Mat mOtsu = otsuProc(mTophat);
16   jLabel5.setIcon(new ImageIcon(toBufferedImage(mOtsu)));
17
18   long startTimeMorfologi = System.nanoTime();
19     DetailObj[] detailObj = new DetailObj[500];
20     for(int i=0;i<500; i++)
21       detailObj[i] = new DetailObj();
22     detailObj = getDetailObj(mOtsu);
23
24     // Layer Morfologi
25     Mat m = mOri.clone();
26     Mat m2 = mOri.clone();
```

```
27      Rect[] arrRectCrop = kandidatLayer1(detailObj, m);
28    long endTimeMorfologi = System.nanoTime();
29    long durationMorfologi = (endTimeMorfologi - startTimeMorfologi)/1000000;
30
31    // Layer Cascade classifier
32    long startTimeCascade = System.nanoTime();
33      doCascade(mOri);
34    long endTimeCascade = System.nanoTime();
35    long durationCascade = (endTimeCascade - startTimeCascade)/1000000;
36
37    // Layer verifikasi
38    long startTimeDuaLayer = System.nanoTime();
39      int hasilKandidat2 = 0;
40      for(int i=0;i<jlhKandidat;i++){
41        if(arrRectCrop[i].x == 0 && arrRectCrop[i].y == 0 && arrRectCrop[i].width == 0 && arrRectCrop[i].height == 0)
42          continue;
43        Mat imCrop = new Mat(m2, arrRectCrop[i]);
44
45        if(verifikasiKandidat(imCrop)){
46          Imgproc.rectangle(m2, new Point(arrRectCrop[i].x, arrRectCrop[i].y), new Point(arrRectCrop[i].x
47              + arrRectCrop[i].width, arrRectCrop[i].y + arrRectCrop[i].height), new Scalar(0, 255, 0));
48          hasilKandidat2 ++;
49        }
50      }
```

```
51    long endTimeDuaLayer = System.nanoTime();
52    long durationDuaLayer = ((endTimeDuaLayer -
      startTimeDuaLayer)/1000000) +
      durationMorfologi;
53
54    System.out.println(fileName + ";" +
      durationMorfologi + ";" + durationCascade +
      ";" + durationDuaLayer);
55    System.out.println("Morfologi: " +
      durationMorfologi + "ms");
56    System.out.println("Cascade: " +
      durationCascade + "ms");
57    System.out.println("Dua layer: " +
      durationDuaLayer + "ms");
58
59    jLabel6.setIcon(new
      ImageIcon(toBufferedImage(m2)));
60    morfologi.setText("Durasi proses morfologi :
      " + durationMorfologi + "ms");
61    cascade.setText("Durasi proses cascade : " +
      durationCascade + "ms");
62    duaLayer.setText("Durasi proses morfologi +
      cascade : " + durationDuaLayer + "ms");
63  }
```

Kode Sumber 4.13 Perintah pada tombol proses



Gambar 4.1 Implementasi halaman utama

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas uji coba dan evaluasi terhadap perangkat lunak yang telah dikembangkan dari implementasi sistem Deteksi Mobil

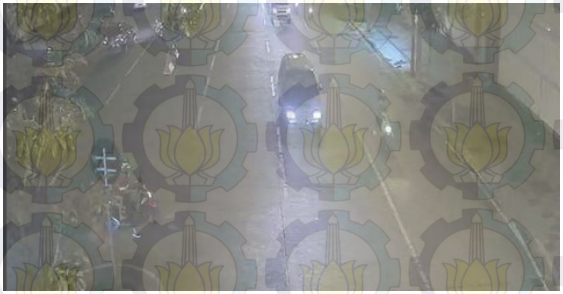
5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Prosesor : Prosesor: Intel® Core™ i7-3517U CPU @ 1.90GHz (4 CPUs) , ~2.4GHz
RAM : 4096 MB
Jenis *Device* : *Notebook*
Sistem Operasi : Microsoft Windows 10

5.2. Data Uji Coba

Data yang digunakan untuk uji coba implementasi sistem Deteksi Mobil ini adalah citra yang diambil dari *CCTV* pada situs pemerintah surabaya dengan ukuran maksimal 360 x 720 piksel dengan *channel* RGB maupun *Grayscale*. Contoh dari citra *CCTV* dapat dilihat pada Gambar 5.1



Gambar 5.1 Data masukan uji coba

5.3. Skenario Uji Coba

Pada subbab ini akan dijelaskan meneganai skenario uji coba yang telah dilakukan. Terdapat beberapa skenario uji coba yang telah dilakukan, diantaranya yaitu:

1. Skenario pengujian 1: Perhitungan nilai *recall*, *precision* dan *running time* dengan hanya menggunakan *layer* operasi morfologi.
2. Skenario pengujian 2: Perhitungan nilai *recall*, *precision* dan *running time* dengan hanya menggunakan *layer cascade classifier*.
3. Skenario pengujian 3: Perhitungan nilai *recall*, *precision* dan *running time* dengan menggunakan operasi morfologi sebagai *layer* pertama dan *cascade classifier* sebagai *layer* kedua.
4. Skenario pengujian 4: Perhitungan nilai *recall*, *precision* dan *running time* menggunakan data *testing* dari *CCTV* yang berbeda menggunakan layer operasi morfologi

5.4. Skenario Pengujian 1: Perhitungan nilai recall, precision dan running time dengan hanya menggunakan layer operasi morfologi

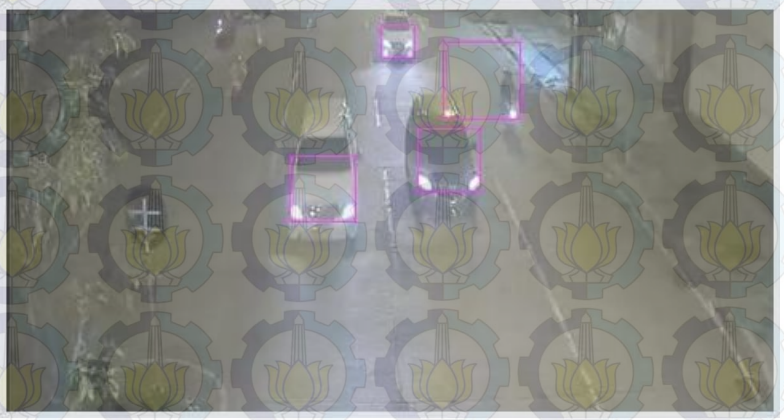
Pada skenario ini dilakukan uji coba sistem deteksi mobil menggunakan *layer* operasi morfologi. Uji coba menggunakan 40 citra *CCTV* dan dilakukan perhitungan nilai *recall* dan nilai *precision* dan juga lama proses. Nilai tersebut dihitung berdasarkan ketentuan berikut:

1. *True positif*: sebuah *tag* dengan objek mobil.
2. *False negatif*: sebuah mobil namun tanpa *tag*.
3. *False positif*: sebuah *tag* namun tanpa objek mobil.

Pada skenario ini menggunakan nilai *threshold* yaitu ukuran *structuring element* pada operasi *Top Hat* yaitu 11, *structuring element* pada operasi *opening* dan *closing* yaitu 5,

maksimal jarak objek pada koordinat *x*, *y* secara berurut yaitu 50 dan 7, dan batas kiri 150, dengan satuan menggunakan ukuran piksel.

Hasil uji coba ini dapat dilihat pada lampiran. Tampak dari 40 citra masukan, nilai rata-rata *recall* dari *layer* ini adalah 96.9% serta untuk nilai rata-rata *precision* adalah 52.8%. Terdapat total 52 mobil terdeteksi dengan benar, namun tidak mendeteksi 4 mobil. Selain itu ada 61 *tag* dengan yang bukan mobil. Selain itu juga dilakukan perhitungan waktu untuk setiap citra yang diproses, didapatkan nilai rata-rata 11.4ms per citra untuk *layer* operasi morfologi. Contoh hasil operasi *layer* dapat dilihat pada Gambar 5.2



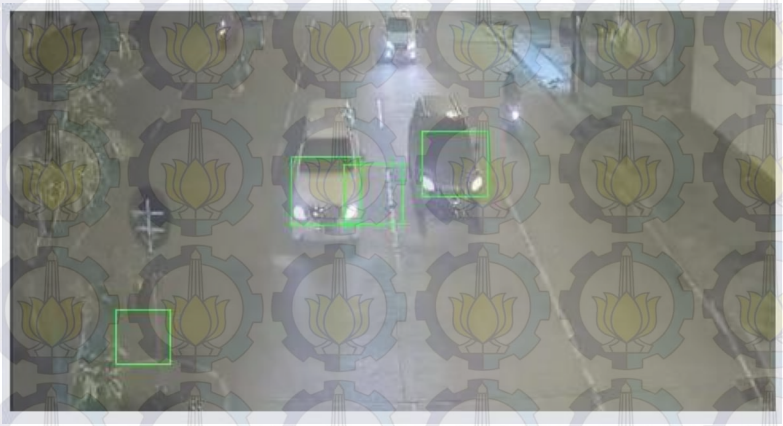
Gambar 5.2 Hasil operasi *layer* morfologi

5.5. Skenario Pengujian 2: Perhitungan nilai recall, precision dan running time dengan hanya menggunakan layer cascade classifier

Pada skenario ini dilakukan uji coba sistem deteksi mobil menggunakan *layer cascade classifier*. Uji coba menggunakan 40 citra *CCTV* dan dilakukan perhitungan nilai *recall* dan nilai *precision* dan juga lama proses. Pada tahap ini menggunakan

classifier hasil *training Adaptive Boosting* dengan jumlah data *training positive* sebanyak 538 dan 1100 data *training negative*.

Hasil uji coba ini dapat dilihat pada lampiran. Tampak dari 40 citra masukan, nilai rata-rata *recall* dari *layer* ini adalah 89.4% serta untuk nilai rata-rata *precision* adalah 55.5%. Terdapat total 45 mobil terdeteksi dengan benar, namun tidak mendeteksi 10 mobil. Selain itu ada 40 *tag* dengan yang bukan mobil. Selain itu juga dilakukan perhitungan waktu untuk setiap citra yang diproses, didapatkan nilai rata-rata 19.7ms per citra untuk *layer cascade classifier*.



Gambar 5.3 Hasil operasi *cascade classifier*

5.6. Skenario Pengujian 3: Perhitungan nilai recall, precision dan running time dengan menggunakan operasi morfologi sebagai layer pertama dan cascade classifier sebagai layer kedua

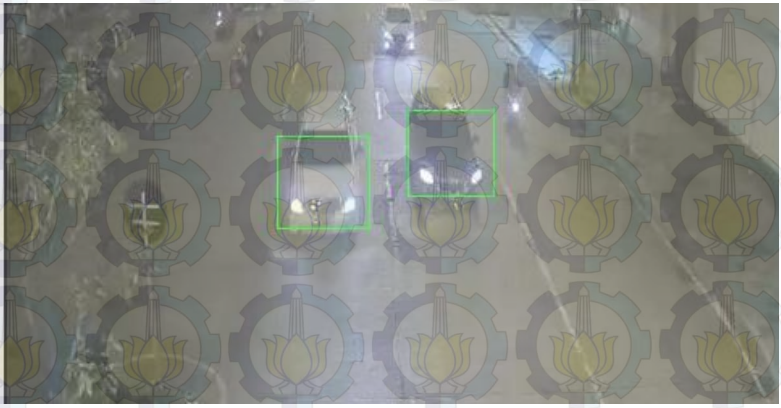
Pada skenario ini dilakukan uji coba sistem deteksi mobil menggunakan dua *layer*, yaitu *layer* operasi morfologi dan *cascade classifier*. Pada tahap ini menggunakan *classifier* hasil *training Adaptive Boosting* dengan jumlah data *training positive* sebanyak 538 dan 1100 data *training negative*. Uji coba

menggunakan 40 citra *CCTV* dan dilakukan perhitungan nilai *recall* dan nilai *precision* dan juga lama proses. Nilai tersebut dihitung berdasarkan ketentuan berikut:

1. *True positif*: sebuah *tag* dengan objek mobil.
2. *False negatif*: sebuah mobil namun tanpa *tag*.
3. *False positif*: sebuah *tag* namun tanpa objek mobil.

Pada skenario ini menggunakan nilai *threshold* yaitu ukuran *structuring element* pada operasi *Top Hat* yaitu 11, *structuring element* pada operasi *opening* dan *closing* yaitu 5, maksimal jarak objek pada koordinat *x*, *y* secara berurut yaitu 50 dan 7, dan batas kiri 150, dengan satuan menggunakan ukuran piksel.

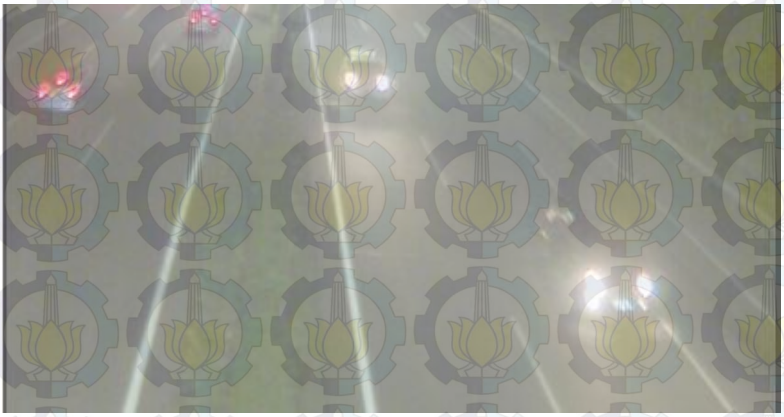
Hasil uji coba ini dapat dilihat pada lampiran. Tampak dari 40 citra masukan, nilai rata-rata *recall* dari *layer* ini adalah 91.5% serta untuk nilai rata-rata *precision* adalah 93.8%. Terdapat total 46 mobil terdeteksi dengan benar, namun tidak mendeteksi 8 mobil. Selain itu ada 6 *tag* dengan yang bukan mobil. Selain itu juga dilakukan perhitungan waktu untuk setiap citra yang diproses, didapatkan nilai rata-rata 20.75ms per citra untuk menggunakan dua *layer*.



Gambar 5.4 Hasil operasi menggunakan *layer* morfologi dan *layer cascade classifier*.

5.7. Skenario Pengujian 4: Perhitungan nilai recall, precision dan running time menggunakan data testing dari CCTV yang berbeda menggunakan layer operasi morfologi

Pada skenario ini dilakukan uji coba sistem deteksi mobil menggunakan data *testing* dari sudut *CCTV* yang berbeda, data *testing* dapat dilihat pada Gambar 5.5. Uji coba akan menghitung nilai *recall* dan nilai *precision*,



Gambar 5.5 Data *testing* dengan sudut *CCTV* yang berbeda

Pada skenario ini menggunakan nilai *threshold* yaitu ukuran *structuring element* pada operasi *Top Hat* yaitu 11, *structuring element* pada operasi *opening* dan *closing* yaitu 1, maksimal jarak objek pada koordinat *x, y* secara berurut yaitu 50 dan 7, dan batas kiri 150, dengan satuan menggunakan ukuran piksel.

Uji coba ini menggunakan 27 citra *CCTV* dengan sudut yang berbeda dari skenario uji coba sebelumnya. Hasil dari uji coba ini adalah 89.6% untuk rata-rata nilai *recall*, dan 97.4% untuk rata-rata nilai *precision*. Contoh hasil keluaran dari uji coba ini dapat dilihat pada Gambar 5.6

Gambar 5.6 Contoh hasil keluaran uji coba dengan data *testing* dengan sudut *CCTV* yang berbeda

5.8. Evaluasi

Dari skenario uji coba yang telah dilakukan dengan beberapa parameter yang telah digunakan dalam uji coba memberikan pengaruh terhadap hasil akurasi kinerja implementasi sistem deteksi mobil ini. Terdapat beberapa hal yang menjadi evaluasi pada sistem deteksi mobil ini, yaitu banyaknya variasi lampu depan mobil sehingga sulitnya untuk menentukan ukuran dari *structuring element* pada operasi *opening* dan *closing*. Selain itu kurangnya data *training* pada sistem ini menjadikan nilai *precision* untuk operasi *cascade* classifier menjadi rendah.

Pada skenario pengujian pertama, data *testing* dengan tingkat intensitas cahaya yang tinggi pada objek seperti pada Gambar 5.7 akan menghasilkan *false alarm*. Hal ini disebabkan karena ukuran *structuring element* yang kecil sehingga hasilnya dapat dilihat pada Gambar 5.8

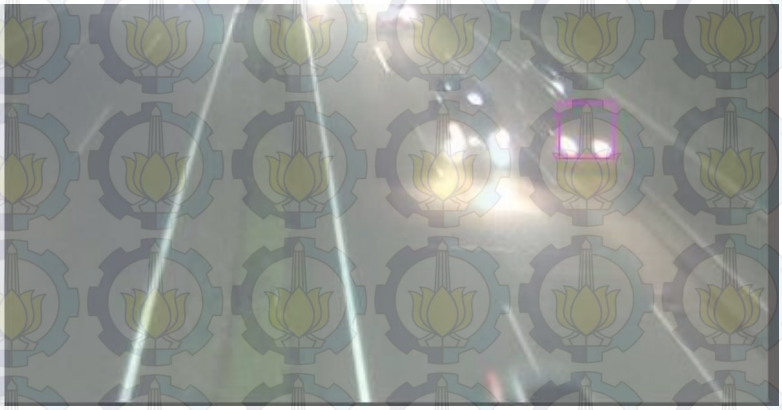
Gambar 5.7 Contoh data *testing* dengan tingkat kecerahan objek yang tinggi.

Gambar 5.8 Hasil operasi morfologi pada Gambar 5.7

Pada Gambar 5.8 dapat dilihat lingkaran merah yang membuat *false alarm* seperti pada Gambar 5.7

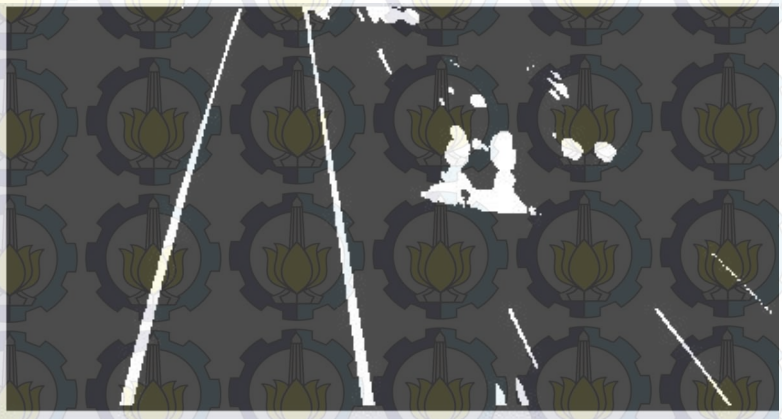
Evaluasi yang didapat pada skenario pengujian ke empat yaitu banyaknya variasi intensitas cahaya lampu depan mobil. Sehingga hal ini membuat sulit untuk menentukan ukuran dari

structuring element pada operasi *opening* dan *closing*. Contoh dapat dilihat pada Gambar 5.9



Gambar 5.9 Contoh kesalahan pendeteksian karena lampu depan mobil sangat terang

Hal ini disebabkan karena ukuran *structuring element* yang digunakan berukuran kecil, sehingga pada operasi *opening* dan *closing* menghasilkan citra seperti pada Gambar 5.10



Gambar 5.10 Hasil operasi Gambar 5.9 menggunakan operasi *opening* dan *closing*

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan Tugas Akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan dating.

6.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Operasi morfologi dalam pendeteksian lampu depan mampu mendeteksi mobil dengan nilai *recall* 96.9%, namun akan menghasilkan banyak *false alarm* dengan nilai 52.8%.
2. Operasi *cascade classifier* mampu mendeteksi dengan nilai *recall* 89.4% yang berarti hanya sedikit jumlah mobil yang tidak terdeteksi, namun menghasilkan banyak *false alarm* dengan nilai *precision* hanya 55.5%. Hal ini dikarenakan jumlah data *training* yang digunakan sedikit.
3. Operasi *cascade classifier* memilki tingkat pendeteksian mobil yang sangat rendah jika menggunakan data *testing* dengan sudut *CCTV* yang berbeda.
4. Dengan menggunakan gabungan dua *layer* operasi morfologi dan *cascade classifier* menjadikan sistem deteksi mobil ini memilki tingkat pendeteksian yang sangat tinggi yaitu 91.5% serta tingat *false alarm* yang sangat rendah yaitu dengan nilai *precision* 93.8%.
5. Operasi dengan menggunakan dua *layer* memiliki kecepatan operasi yang hampir sama dibandingkan dengan operasi *cascade classifier*, dengan kecepatan rata-rata yaitu 20.75ms untuk menggunakan dua *layer*, sedangkan 19.7ms dengan menggunakan *cascade classifier*.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Perlu dilakukan uji coba yang lebih banyak lagi.
2. Jumlah data *training* harus lebih banyak lagi agar sistem yang dibangun lebih *robust*.
3. Menggunakan model yang berbeda untuk sudut *CCTV* yang berbeda.

DAFTAR PUSTAKA

[1] B. Y. Pratama, "saintek.uin-malang.ac.id," [Online]. Available: http://saintek.uin-malang.ac.id/Mirror/ilmukomputer/Batra-Operasi-Morfologi-Pada-Citra-Biner.pdf.

[2] D. N. Rahmah, "digilib.its.ac.id," [Online]. Available: http://digilib.its.ac.id/public/ITS-Undergraduate-16849-5107100014-paper-1pdf.pdf. [Diakses 12 1 2016].

[3] Y. Freund, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence,* 1999.

[4] D. Gerónimo, "http://www.cvc.uab.es/," 2009. [Online]. Available: http://www.cvc.uab.es/adas/site/userfiles/files/haarfeatures.pdf.

[5] V. a. Jones, "Rapid object detection using a boosted cascade of simple features," *Computer Vision and Pattern Recognition,* 2001.

[6] OpenCV, "OpenCV 2.4.9.0 documentation," 21 April 2014. [Online]. Available: http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html. [Diakses 30 September 2014].

[7] "Scale Factor," [Online]. Available: http://www.icoachmath.com/math_dictionary/scale_factor.html. [Diakses 5 Januari 2016].

[8] R. Akbar, "unikom.ac.id," [Online]. Available: http://elib.unikom.ac.id/files/disk1/593/jbptunikompp-gdl-rizqiakbar-29637-9-unikom_r-i.pdf. [Diakses 12 1 2016].

[9] M. Rezaei, "www.cs.auckland.ac.nz," [Online]. Available: https://www.cs.auckland.ac.nz/~m.rezaei/Tutorials/Creatin

g_a_Cascade_of_Haar-Like_Classifiers_Step_by_Step.pdf. [Accessed 12 1 2016].

[10] OpenCV, "opencv.org," [Online]. Available: http://docs.opencv.org/java/3.0.0/. [Diakses 12 1 2016].

[11] W. Wang, "A Two-Layer Night-time Vehicle Detector," *Digital Image Computing: Techniques and Applications*, 2009.

[12] D. Jacobs, "Correlation and Convolution," [Online]. Available: http://www.cs.umd.edu/~djacobs/CMSC426/Convolution.pdf. [Diakses 5 January 2015].

[13] F. C. Crow, "Summed-area tables for texture mapping," *Computer Graphics*, vol. 18, p. 1984.

[14] OpenCV, "opencv.org," [Online]. Available: http://docs.opencv.org/3.0-beta/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=connectedcomponents#connectedcomponents. [Diakses 12 1 2016].

LAMPIRAN

Lampiran A. 1 Hasil uji coba deteksi mobil dengan menggunakan *layer* operasi morfologi

No	Nama File	Morfologi			Time	Recall	Precision
		TP	FN	FP	Morfologi	Morfologi	Morfologi
1	1.jpg	1	0	1	14	100.0	50.0
2	2.jpg	1	0	1	15	100.0	50.0
3	3.jpg	1	0	2	15	100.0	33.3
4	4.jpg	0	0	0	14	100.0	100.0
5	5.jpg	1	0	2	14	100.0	33.3
6	6.jpg	0	0	2	11	100.0	0.0
7	7.jpg	2	0	2	13	100.0	50.0
8	8.jpg	1	0	0	9	100.0	100.0
9	9.jpg	0	0	2	12	100.0	0.0
10	10.jpg	0	0	0	9	100.0	100.0
11	11.jpg	1	0	0	9	100.0	100.0
12	12.jpg	1	2	1	9	33.3	50.0
13	13.jpg	2	0	4	10	100.0	33.3
14	14.jpg	2	0	3	9	100.0	40.0
15	15.jpg	2	0	0	10	100.0	100.0
16	16.jpg	1	0	2	11	100.0	33.3
17	17.jpg	1	0	1	23	100.0	50.0
18	18.jpg	0	0	1	10	100.0	0.0
19	19.jpg	1	0	1	10	100.0	50.0
20	20.jpg	1	0	0	12	100.0	100.0
21	21.jpg	2	0	1	11	100.0	66.7
22	22.jpg	0	0	2	10	100.0	0.0
23	23.jpg	1	0	3	14	100.0	25.0
24	24.jpg	3	1	1	11	75.0	75.0
25	25.jpg	0	0	2	10	100.0	0.0
26	26.jpg	3	0	3	12	100.0	50.0
27	27.jpg	1	0	0	10	100.0	100.0
28	28.jpg	2	1	1	11	66.7	66.7
29	29.jpg	1	0	0	10	100.0	100.0

30	30.jpg	2	0	2	11	100.0	50.0
31	31.jpg	2	0	2	10	100.0	50.0
32	32.jpg	1	0	0	9	100.0	100.0
33	33.jpg	3	0	1	11	100.0	75.0
34	34.jpg	2	0	3	10	100.0	40.0
35	35.jpg	0	0	2	11	100.0	0.0
36	36.jpg	1	0	2	11	100.0	33.3
37	37.jpg	1	0	4	11	100.0	20.0
38	38.jpg	3	0	0	15	100.0	100.0
39	39.jpg	3	0	2	11	100.0	60.0
40	40.jpg	2	0	5	11	100.0	28.6
Rata-rata					11.475	96.9	52.8

Lampiran A. 2 Hasil uji coba deteksi mobil dengan menggunakan *layer cascade classifier*

No	Nama File	*Cascade classifier*			Time (ms)	*Recall* (%)	*Precision* (%)
		TP	FN	FP	*Cascade*	*Cascade*	*Cascade*
1	1.jpg	1	0	1	34	100.0	50.0
2	2.jpg	1	0	1	23	100.0	50.0
3	3.jpg	1	0	1	28	100.0	50.0
4	4.jpg	0	0	1	25	100.0	0.0
5	5.jpg	1	0	0	17	100.0	100.0
6	6.jpg	0	0	1	17	100.0	0.0
7	7.jpg	2	0	2	26	100.0	50.0
8	8.jpg	1	0	1	16	100.0	50.0
9	9.jpg	0	0	1	16	100.0	0.0
10	10.jpg	0	0	1	21	100.0	0.0
11	11.jpg	1	0	0	18	100.0	100.0
12	12.jpg	1	2	0	15	33.3	100.0

13	13.jpg	2	0	0	16	100.0	100.0
14	14.jpg	1	1	1	16	50.0	50.0
15	15.jpg	2	0	1	16	100.0	66.7
16	16.jpg	1	0	0	17	100.0	100.0
17	17.jpg	0	1	2	17	0.0	0.0
18	18.jpg	0	0	0	16	100.0	100.0
19	19.jpg	1	0	0	18	100.0	100.0
20	20.jpg	0	0	0	19	100.0	100.0
21	21.jpg	2	0	2	17	100.0	50.0
22	22.jpg	0	0	1	17	100.0	0.0
23	23.jpg	1	0	4	22	100.0	20.0
24	24.jpg	3	1	0	17	75.0	100.0
25	25.jpg	0	0	1	17	100.0	0.0
26	26.jpg	3	0	1	20	100.0	75.0
27	27.jpg	1	0	1	16	100.0	50.0
28	28.jpg	2	1	2	16	66.7	50.0
29	29.jpg	1	0	3	16	100.0	25.0
30	30.jpg	2	0	2	16	100.0	50.0
31	31.jpg	2	0	0	16	100.0	100.0
32	32.jpg	1	0	1	15	100.0	50.0
33	33.jpg	2	1	1	16	66.7	66.7
34	34.jpg	2	0	1	16	100.0	66.7
35	35.jpg	0	0	1	21	100.0	0.0
36	36.jpg	1	0	0	16	100.0	100.0
37	37.jpg	1	0	1	16	100.0	50.0
38	38.jpg	2	1	2	19	66.7	50.0
39	39.jpg	2	1	2	71	66.7	50.0
40	40.jpg	1	1	0	17	50.0	100.0

Rata-rata		45	10	40	19.7	89.4	55.5

Lampiran A. 3 Hasil uji coba deteksi mobil dengan menggunakan *layer* operasi morfologi dan *cascade classifier*

No	Nama File	Dua *Layer*			*Time* (ms)	*Recall* (%)	*Precision* (%)
		TP	FN	FP	2 *Layer*	2 *Layer*	2 *Layer*
1	1.jpg	1	0	0	19	100.0	100.0
2	2.jpg	1	0	1	23	100.0	50.0
3	3.jpg	1	0	1	29	100.0	50.0
4	4.jpg	0	0	0	18	100.0	100.0
5	5.jpg	1	0	0	22	100.0	100.0
6	6.jpg	0	0	0	16	100.0	100.0
7	7.jpg	2	0	0	27	100.0	100.0
8	8.jpg	0	1	0	12	0.0	100.0
9	9.jpg	0	0	0	18	100.0	100.0
10	10.jpg	0	0	0	9	100.0	100.0
11	11.jpg	1	0	0	12	100.0	100.0
12	12.jpg	1	2	0	15	33.3	100.0
13	13.jpg	2	0	0	28	100.0	100.0
14	14.jpg	2	0	1	29	100.0	66.7
15	15.jpg	2	0	0	16	100.0	100.0
16	16.jpg	1	0	0	22	100.0	100.0
17	17.jpg	0	0	0	28	100.0	100.0
18	18.jpg	0	0	0	13	100.0	100.0
19	19.jpg	1	0	0	16	100.0	100.0
20	20.jpg	0	0	0	15	100.0	100.0

21	21.jpg	2	0	0	21	100.0	100.0
22	22.jpg	0	0	0	16	100.0	100.0
23	23.jpg	1	0	1	28	100.0	50.0
24	24.jpg	3	1	0	24	75.0	100.0
25	25.jpg	0	0	0	15	100.0	100.0
26	26.jpg	3	0	0	39	100.0	100.0
27	27.jpg	1	0	0	13	100.0	100.0
28	28.jpg	2	1	0	21	66.7	100.0
29	29.jpg	1	0	0	13	100.0	100.0
30	30.jpg	2	0	0	20	100.0	100.0
31	31.jpg	2	0	1	24	100.0	66.7
32	32.jpg	1	0	0	12	100.0	100.0
33	33.jpg	2	1	0	23	66.7	100.0
34	34.jpg	2	0	1	25	100.0	66.7
35	35.jpg	0	0	0	18	100.0	100.0
36	36.jpg	1	0	0	19	100.0	100.0
37	37.jpg	1	0	0	25	100.0	100.0
38	38.jpg	3	0	0	25	100.0	100.0
39	39.jpg	2	1	0	28	66.7	100.0
40	40.jpg	1	1	0	34	50.0	100.0
Rata-rata		46	8	6	20.75	91.5	93.8

Lampiran A. 4 Hasil uji coba deteksi mobil dengan menggunakan *layer* operasi morfologi pada citra *CCTV* dengan sudut yang berbeda

No	Nama File	Morfologi			Time (ms)	Recall (%)	Precision (%)
		TP	FN	FP			
1	1.jpg	2	0	0	19	100.0	100.0
2	2.jpg	1	0	0	23	100.0	100.0
3	3.jpg	1	0	0	29	100.0	100.0
4	4.jpg	3	1	0	18	75.0	100.0
5	5.jpg	1	0	0	22	100.0	100.0
6	6.jpg	1	0	0	16	100.0	100.0
7	7.jpg	1	0	0	27	100.0	100.0
8	8.jpg	3	2	0	12	60.0	100.0
9	9.jpg	1	0	0	18	100.0	100.0
10	10.jpg	3	0	0	9	100.0	100.0
11	11.jpg	1	1	0	12	50.0	100.0
12	12.jpg	2	0	0	15	100.0	100.0
13	13.jpg	1	0	0	28	100.0	100.0
14	14.jpg	2	1	0	29	66.7	100.0
15	15.jpg	1	0	0	16	100.0	100.0
16	16.jpg	1	0	0	22	100.0	100.0
17	17.jpg	1	0	0	28	100.0	100.0
18	18.jpg	1	1	0	13	50.0	100.0
19	19.jpg	3	0	0	16	100.0	100.0
20	20.jpg	4	0	1	15	100.0	80.0
21	21.jpg	1	0	0	21	100.0	100.0
22	22.jpg	3	0	0	16	100.0	100.0
23	23.jpg	1	0	1	28	100.0	50.0

24	24.jpg	2	1	0	24	66.7	100.0
25	25.jpg	2	0	0	15	100.0	100.0
26	26.jpg	1	0	0	39	100.0	100.0
27	27.jpg	1	1	0	13	50.0	100.0
Rata-rata		45	8	2	20.75	89.6	97.4

Lampiran A. 5

```
1  private void initComponents() {
2    jLabel1 = new javax.swing.JLabel();
3    jLabel2 = new javax.swing.JLabel();
4    jLabel3 = new javax.swing.JLabel();
5    jLabel4 = new javax.swing.JLabel();
6    jLabel5 = new javax.swing.JLabel();
7    jLabel6 = new javax.swing.JLabel();
8    filenameTextField = new
javax.swing.JTextField();
9    pilihButton = new javax.swing.JButton();
10   prosesButton = new javax.swing.JButton();
11   morfologi = new javax.swing.JTextField();
12   cascade = new javax.swing.JTextField();
13   duaLayer = new javax.swing.JTextField();
14
15
setDefaultCloseOperation(javax.swing.WindowCon
stants.EXIT_ON_CLOSE);
16
17   jLabel1.setText("jLabel1");
18
19   jLabel2.setText("jLabel2");
```

```
20
21      jLabel3.setText("jLabel3");
22
23      jLabel4.setText("jLabel4");
24
25      jLabel5.setText("jLabel5");
26
27      jLabel6.setText("jLabel6");
28
29      filenameTextField.setText("Lokasi gambar");
30
31      pilihButton.setText("Pilih gambar");
32      pilihButton.addActionListener(new
    java.awt.event.ActionListener() {
33          public void
    actionPerformed(java.awt.event.ActionEvent
    evt) {
34              pilihButtonActionPerformed(evt);
35          }
36      });
37
38      prosesButton.setText("Proses");
39      prosesButton.addActionListener(new
    java.awt.event.ActionListener() {
40          public void
    actionPerformed(java.awt.event.ActionEvent
    evt) {
41              prosesButtonActionPerformed(evt);
42          }
43      });
44
45      morfologi.setText("Durasi proses morfologi :
    ");
```

```
46
47      cascade.setText("Durasi proses cascade : ");
48      cascade.addActionListener(new
        java.awt.event.ActionListener() {
49          public void
        actionPerformed(java.awt.event.ActionEvent
        evt) {
50              cascadeActionPerformed(evt);
51          }
52      });
53
54      duaLayer.setText("Durasi proses morfologi +
        cascade :");
55      duaLayer.addActionListener(new
        java.awt.event.ActionListener() {
56          public void
        actionPerformed(java.awt.event.ActionEvent
        evt) {
57              duaLayerActionPerformed(evt);
58          }
59      });
60
61      javax.swing.GroupLayout layout = new
        javax.swing.GroupLayout(getContentPane());
62      getContentPane().setLayout(layout);
63      layout.setHorizontalGroup(
64
        layout.createParallelGroup(javax.swing.GroupLa
        yout.Alignment.LEADING)
65          .addGroup(layout.createSequentialGroup()
66              .addContainerGap()
67
        .addGroup(layout.createParallelGroup(javax.swi
        ng.GroupLayout.Alignment.LEADING)
68
        .addGroup(layout.createSequentialGroup()
```

```
69                  .addGap(0, 0, Short.MAX_VALUE)
70  .addGroup(layout.createParallelGroup(javax.swi
    ng.GroupLayout.Alignment.LEADING)
71                  .addComponent(jLabel1,
    javax.swing.GroupLayout.PREFERRED_SIZE, 500,
    javax.swing.GroupLayout.PREFERRED_SIZE)
72                  .addComponent(jLabel2,
    javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.PREFERRED_SIZE, 500,
    javax.swing.GroupLayout.PREFERRED_SIZE)))
73  .addGroup(layout.createSequentialGroup()
74  .addGroup(layout.createParallelGroup(javax.swi
    ng.GroupLayout.Alignment.LEADING)
75  .addGroup(layout.createSequentialGroup()
76                  .addComponent(filenameTextField,
    javax.swing.GroupLayout.PREFERRED_SIZE, 328,
    javax.swing.GroupLayout.PREFERRED_SIZE)
77  .addPreferredGap(javax.swing.LayoutStyle.Compo
    nentPlacement.UNRELATED)
78                  .addComponent(pilihButton)
79  .addPreferredGap(javax.swing.LayoutStyle.Compo
    nentPlacement.RELATED)
80                  .addComponent(prosesButton))
81                  .addComponent(morfologi,
    javax.swing.GroupLayout.PREFERRED_SIZE, 300,
    javax.swing.GroupLayout.PREFERRED_SIZE))
82                  .addGap(0, 0, Short.MAX_VALUE)))
83          .addGap(18, 18, 18)
84  .addGroup(layout.createParallelGroup(javax.swi
    ng.GroupLayout.Alignment.LEADING)
85                  .addComponent(jLabel4,
    javax.swing.GroupLayout.PREFERRED_SIZE, 500,
    javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
86          .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 500,
javax.swing.GroupLayout.PREFERRED_SIZE)
87          .addComponent(cascade,
javax.swing.GroupLayout.PREFERRED_SIZE, 301,
javax.swing.GroupLayout.PREFERRED_SIZE))
88
.addPreferredGap(javax.swing.LayoutStyle.Compo
nentPlacement.RELATED)
89
.addGroup(layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)
90          .addComponent(duaLayer,
javax.swing.GroupLayout.PREFERRED_SIZE, 301,
javax.swing.GroupLayout.PREFERRED_SIZE)
91          .addComponent(jLabel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 500,
javax.swing.GroupLayout.PREFERRED_SIZE)
92          .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 500,
javax.swing.GroupLayout.PREFERRED_SIZE))
93        .addContainerGap())
94    );
95    layout.setVerticalGroup(
96
layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)
97
.addGroup(javax.swing.GroupLayout.Alignment.TR
AILING, layout.createSequentialGroup()
98        .addContainerGap()
99
.addGroup(layout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.BASELINE)
100         .addComponent(prosesButton)
101         .addComponent(pilihButton)
102         .addComponent(filenameTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
103         .addGap(18, 18, 18)
104 .addGroup(layout.createParallelGroup(javax.swi
    ng.GroupLayout.Alignment.LEADING)
105 .addGroup(layout.createSequentialGroup()
106             .addComponent(jLabel5,
    javax.swing.GroupLayout.PREFERRED_SIZE, 270,
    javax.swing.GroupLayout.PREFERRED_SIZE)
107 .addPreferredGap(javax.swing.LayoutStyle.Compo
    nentPlacement.RELATED)
108             .addComponent(jLabel6,
    javax.swing.GroupLayout.PREFERRED_SIZE, 270,
    javax.swing.GroupLayout.PREFERRED_SIZE))
109 .addGroup(layout.createParallelGroup(javax.swi
    ng.GroupLayout.Alignment.TRAILING)
110 .addGroup(javax.swing.GroupLayout.Alignment.LE
    ADING, layout.createSequentialGroup()
111             .addComponent(jLabel3,
    javax.swing.GroupLayout.PREFERRED_SIZE, 270,
    javax.swing.GroupLayout.PREFERRED_SIZE)
112 .addPreferredGap(javax.swing.LayoutStyle.Compo
    nentPlacement.RELATED)
113             .addComponent(jLabel4,
    javax.swing.GroupLayout.PREFERRED_SIZE, 270,
    javax.swing.GroupLayout.PREFERRED_SIZE))
114 .addGroup(layout.createSequentialGroup()
115             .addComponent(jLabel1,
    javax.swing.GroupLayout.PREFERRED_SIZE, 270,
    javax.swing.GroupLayout.PREFERRED_SIZE)
116 .addPreferredGap(javax.swing.LayoutStyle.Compo
    nentPlacement.RELATED)
117             .addComponent(jLabel2,
    javax.swing.GroupLayout.PREFERRED_SIZE, 270,
    javax.swing.GroupLayout.PREFERRED_SIZE))))
118         .addGap(18, 18, 18)
```

```
119 
    .addGroup(layout.createParallelGroup(javax.swi
    ng.GroupLayout.Alignment.LEADING)
120         .addComponent(morfologi,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)
121         .addComponent(cascade,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)
122         .addComponent(duaLayer,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))
123       .addContainerGap(15, Short.MAX_VALUE))
124   );
125 
126     pack();
127 }// </editor-fold>
```

Kode Sumber 4.14

BIODATA PENULIS

Penulis, Muhammad Aunorafiq M usa, lahir di Pangkajene dan Kepulauan, pada tanggal 14 Juli 1995. Penulis menempuh pendidikan sekolah dasar di SDN 1 Pekkai, Barru. Melanjutkan pendidikan sekolah menengah pertama di SMPN 12 Makassar dan selanjutnya di SMAN 15 Makassar. Selanjutnya penulis melanjutkan pendidikan sarjana di Jurusan Teknik Informatika, Fakultas Teknologi dan Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif menjadi administrator di Laboratorium Pemrograman 2 Teknik Informatika dan aktif dalam organisasi tingkat Jurusan sebagai wakil ketua himpunan, dan tingkat Fakultas sebagai staff.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Komputasi Cerdas dan Visi (KCV) dan memiliki ketertarikan di bidang komputer visi dan juga manajemen SDM. Penulis dapat dihubungi melalui surel: raffi.musa07@gmail.com.